Induction of Predictive Models for Dynamical Systems via Data Mining

Danny Van Welden

Proefschrift voorgedragen tot het behalen van het doctoraat
in de wetenschappen, richting informatica

Promotoren: Prof. Dr. H. De Meyer
Prof. Dr. ir. G.C. Vansteenkiste

Academiejaar 1999-2000

Acknowledgements

The more you learn the less you know

I am very grateful to Professor G. Vansteenkiste for providing me with my first job and for offering me many opportunities to enhance my knowledge in different domains. In addition, I owe a great debt of gratitude to Professor F. Cellier and to Professor E.J.H. Kerckhoffs, who helped me when I needed them and who provided me useful feedback with regard to this thesis. I am grateful to Prof. H. De Meyer who helped me unconditionally during the finalisation of my work. Moreover, I would like to thank all the above professors for their belief in me (often more than I did) and for their encouragements.
I would also like to thank the colleagues at my work for the pleasant working environment they helped to create. I like to thank especially Phillippe Geril for his revision of my English writing and Annie Gevaert who helped me in all the administrative chores that had to be done. Last, but not least, this thesis would not have been possible without the patience of my wife Gerda and my children Laurenz and Maurijn, who had to endure my absence when I was away for yet another course. Gerda's support and understanding has been responsible for this thesis in more ways than she could ever imagine.

Danny Florent Van Welden

Gent, November 22, 1999.

Contents

PART II: A Data Mining Approach to Identification of Dynamical Systems

CHAPTER 5: FROM KNOWLEDGE DISCOVERY IN DATABASES TO CLASSIFICATION

CHAPTER 6: CLASSIFICATION AND REGRESSION TREES

CHAPTER 7: RELATIONSHIP BETWEEN GENERAL SYSTEM THEORY AND KNOWLEDGE DISCOVERY IN DATABASES VIA META-MODELLING

CHAPTER 8: THE USE OF CLASSIFICATION AND REGRESSION TREES FOR SAPS

Appendices

Introduction

The thesis describes two fundamental contributions towards the induction of predictive models for dynamical systems, based on a pattern recognition approach. One contribution involves the enrichment of an experimental software tool for non-parametric system identification, called SAPS [Cellier 1991]. The other concerns the establishment of a promising link of SAPS' underlying framework, called GSPS [Klir 1985], with a relatively young research domain that has the necessary methodology to aid in GSPS' problem solving. The contributions proposed in this thesis help to overcome the limited power of SAPS to the identification of complex black-box systems. As the thesis demonstrates, the most promising perspectives of achieving a good predictive model, is via the use of data mining methods.

The first contribution remains entirely in the domain of general system theory (GST), from which SAPS originates. The second contribution implies a total paradigm shift that goes back to the re-use of basic principles in GSPS. The second modification transcends the boundaries of GST towards the domain of Knowledge Discovery in Databases (KDD); system identification approach in GST is hereby considered as supervised learning applied to dynamical systems. Consequently, this thesis bridges a gap between the two domains. The GST domain deals with the identification and modelling of dynamical systems, while the KDD domain tries to identify useful and non-trivial patterns from observed, mostly static, data.

In general system theory, many methodologies exist, but one of them, called General System Problem Solving (GSPS) [Klir 1985], is of particular importance in the sub-domain of system identification, because it forms the backbone of SAPS. A central principle of GSPS that maps observed input-output data to a state space via a time-invariant dependency pattern forms the basis of a transformation that rephrases the system identification problem in GST to a data mining problem in KDD. In the latter domain, a plethora of existing techniques can be used to solve the deficiency of SAPS, towards the identification of complex systems, indirectly. This results in an extremely powerful approach to system identification for large complex systems where many variables come into play.

This thesis provides an overview of the two domains involved in two separate parts. Each part moves gradually from the general underlying framework towards a specific implementation tool. The first two chapters of each part are based on current literature. The remaining two chapters in each part are new contributions of the dissertation. A summary of the major contributions of this thesis is found in the conclusion at the end.

Part I: A System Theoretic Approach

Chapter 1 describes the underlying framework for the non-parametric system identification approach in the domain of systems theory. It balances a formalised approach with a more descriptive approach. The chapter moves from General System Theory towards General System Problem Solving. The latter permits non-parametric identification of black-box systems via an inductive approach based on pattern-recognition. The General System Problem Solving framework is elaborated and more formalisations are given. They help to place the SAPS tool, which is described in chapter 2, in the right context. The very important concept of a 'mask'

is introduced and formalised in chapter 1. Throughout the chapter, some initial assumptions with regard to the goal of the dissertation are highlighted.
Chapter 2 elaborates on a popular version of SAPS, which is developed by Cellier [1991] and is called SAPS-II. It has proven to have a good potential for practical applications and as such it will serve as a reference for the research developed in this dissertation. Although SAPS-II is based on GSPS, some methodological differences and restrictions apply that are explained in this chapter. Attaching a quality to a mask provides the possibility to compare masks and to search for the best one. Two existing forecasting methods, which are compared with the new methods introduced in chapter 8, are explained.
Chapter 3 establishes a link between problem types in SAPS and in hidden Markov models via a rigorous formalisation of 'state-transition' matrix construction. It shows that correlation in the system memory does not play a role for SAPS and that induction problems are similar in SAPS as in hidden Markov models [Van Welden 20xx].
The first part of chapter 4 is theoretical and concentrates on the introduction and justification of a newly introduced sub-optimal approach. The order complexity of the subsequent algorithm is not exponential as it is for the existing optimal one, but polynomial. This results in the possibility of tackling systems that are more complex [Van Welden and Vansteenkiste 1996]. The second part of chapter 4 gives a brief overview of the newly designed tool that supports the sub-optimal mask search. This tool, which is called SAPS-ST, is intended as a prototype to explore new model construction techniques [Van Welden and Vansteenkiste 1994]. Compatibility between the new software tool, SAPS-ST, and the existing tool SAPS-II is not described in this dissertation. The reader is referred to [Van Welden 1999].

Part II: A Data Mining Approach

Chapter 5 provides an overview of the emerging domain of Knowledge Discovery in Databases (KDD) and data mining. It follows the same structure as chapter 1. The field of machine learning is briefly touched upon, because some concepts help in understanding the material chapter 5 contains. It shows that system identification by SAPS belongs to a supervised learning paradigm. The latter encompasses classification and regression. A formalisation of classification is undertaken and tree classifiers are introduced at the end of this chapter. Chapter 5 is indispensable for showing the plethora of methods that can be applied when doing system-identification via supervised learning.
Chapter 6 concentrates on tree classifiers and shows their underlying principles. This chapter supplies the knowledge to understand chapter 8 where the use of tree classifiers for SAPS is demonstrated. It elaborates on a specific statistical approach for classification and regression trees on which the software tool CART© resides.
Chapter 7 explicitly establishes the link between GST and KDD. GSPS can be seen as a data-mining approach, and, consequently, it can be embedded in KDD [Van Welden et al. 1998]. Relationships and emphasis shifts are explained at different levels of abstraction. A sharper focus to directed systems (in GST) and to supervised learning (in KDD) is done. The resulting paradigm shift for SAPS is outlined.
Chapter 8 illustrates how to apply data mining to SAPS with the aid of tree classifiers. The latter are chosen because they give comprehensible models and because they have much in common with the extension made in chapter 4. The advantages of using tree classifiers are shown for recoding and for the handling of missing values [Van Welden et al. 1999]. An extra benefit is the ranking of variable importance. In this chapter, the implementation of the nearest neighbour algorithm in SAPS-II is modified and simplified. The new nearest neighbour methods should be seen in the context of data mining.

© CART is a product from Salford Systems Inc., see www.salford-systems.com

Appendices contain examples that illustrate more than one aspect of the data mining approach. Appendix A and B contain synthetic examples, while appendix C and D contain real-world examples. Each appendix emphasises different aspects by the example it contains.

Relevant articles by the author

[Van Welden et al. 1991]	Van Welden D., Verweij D., Vansteenkiste G.C., "*A Proposal for Incorporating Heuristic Knowledge in a Multifaceted System*", Proceedings of EUROCAST 91 - 2nd International Workshop on Computer Aided Systems Theory, Krems (Wachau), Austria, April 15-19, 1991, p. 295-306.
[Van Welden and Vansteenkiste 1992]	Van Welden D., Vansteenkiste G.C., "*A Mixed Deductive-Inductive Approach to Model Recognition*", Proceedings of the 1992 European Simulation Multiconference, York, UK, June 1-3, p. 112-118, 1992.
[Van Welden and Vansteenkiste 1994]	Van Welden D., Vansteenkiste G.C., "*SAPS-ST: A Testbed For Incremental Research on GSPS"*, Proceedings of the 1994 European Simulation Multiconference, Barcelona, Spain, June 1-3, p. 507-513, 1994
[Van Welden and Vansteenkiste 1996]	Van Welden D., Vansteenkiste G.C., "*Sub-Optimal Mask Search in SAPS"*, International Journal of General Systems, vol. 24, 1-2, p. 137-150, 1996.
[Van Welden 1998]	Van Welden D., Tree Classifiers as Data Mining Tools. MSc. Thesis, Catholic University of Leuven, Belgium, 1998.
[Van Welden et al. 1998]	Van Welden D., Kerckhoffs E.J.H., Vansteenkiste G.C., *Extending a Fuzzy Inductive Reasoner with Classification Procedures*, Simulation Technology: Science and Art, Proc. 10th European Simulation Symposium and Exhibition, ESS 98, ed. A. Bargiela and E. Kerckhoffs, Nottingham, October 26-28, UK, p. 111- 116, 1998.
[Van Welden 1999]	Van Welden D., Compatibility of SAPS-ST and SAPS-II (with user-manual), Technical Report, 1999.
[Van Welden et al. 1999]	Van Welden D., Kerckhoffs E.J.H., Vansteenkiste G.C., *"Automatic Recoding in a Fuzzy Inductive Reasoner"*, ICQFN'99, Warsaw, June 1-4, p. 348-354, 1999.
[Van Welden 20xx]	Van Welden D., "*Formalizing Candidate Model Construction in SAPS With Hidden Markov Models"*, International Journal of General Systems, accepted.

Other references

[Cellier 1991]	Cellier F.E., "*Continuous System Modelling*", Springer-Verlag, New York, 1991.
[Klir 1985]	Klir G.J., Architecture of Systems Problem Solving, Plenum Press, 1985.

Figures that clarify the arrangement of the thesis

Figure 1 shows the decreasing level of abstraction.

Figure 2 shows the main parts of this thesis with their chapters. Chapters that are new contributions of the thesis are depicted in bold

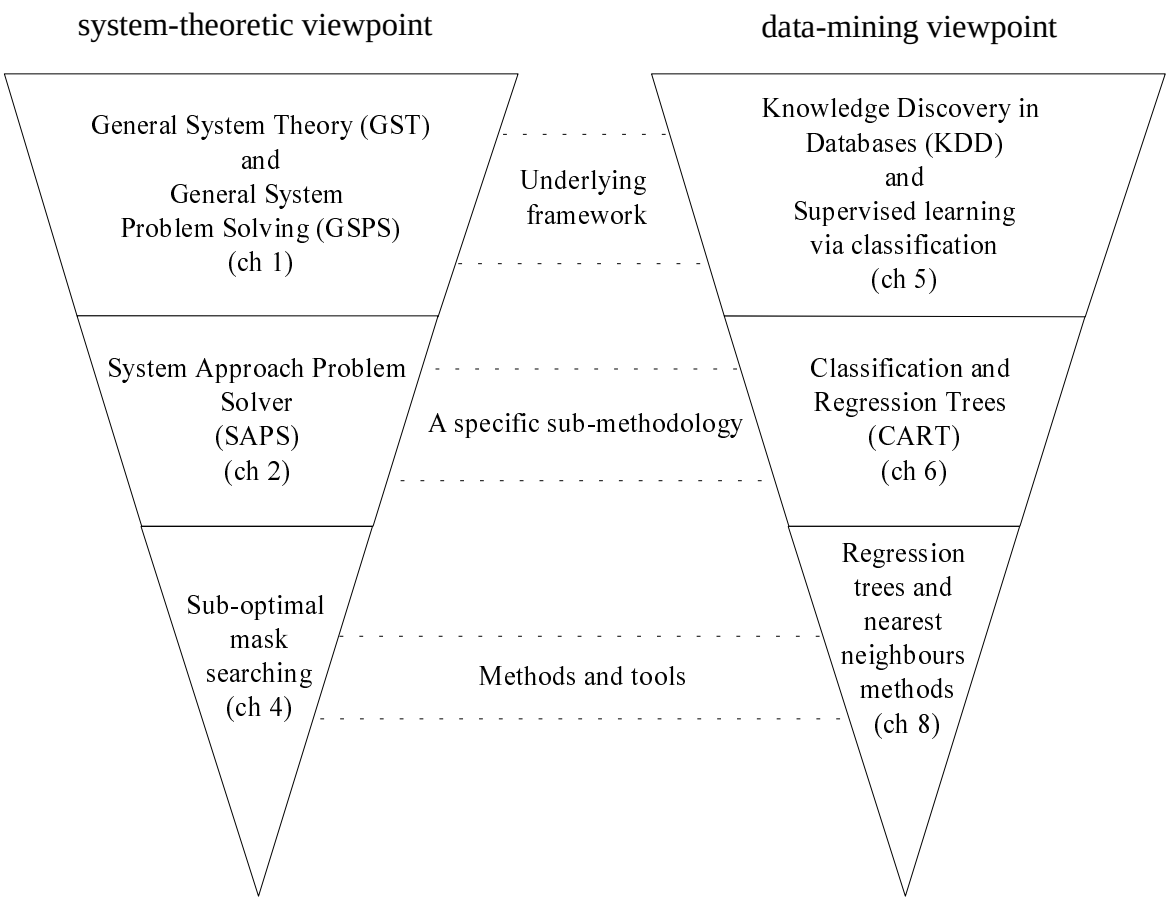


Figure 1 : The stepwise refined structure of this thesis

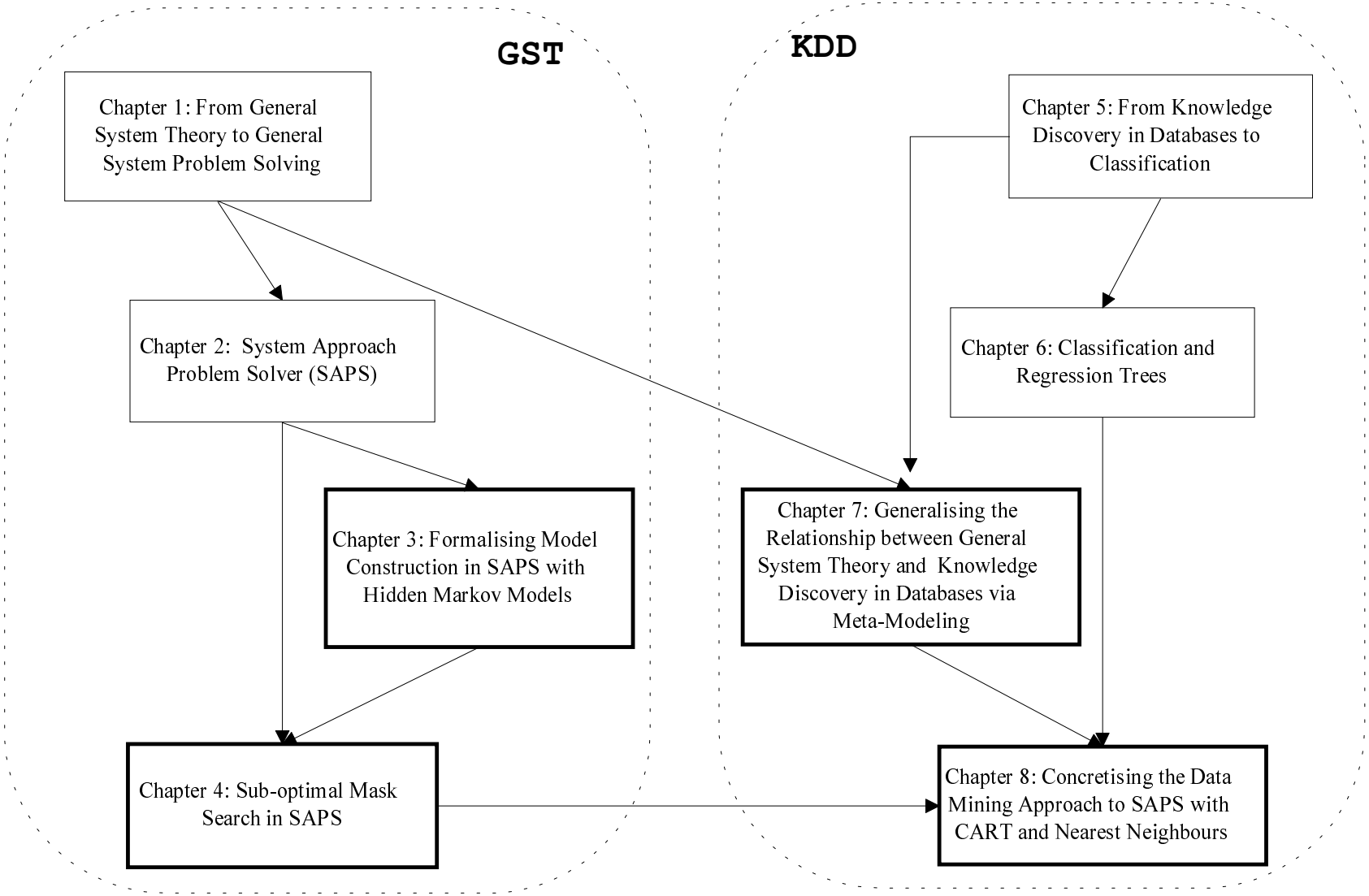


Figure 2 : Overview of the thesis (bold rectangles cover original contributions by the author)

PART I

A System Theoretic Approach to Identification of Dynamical Systems

Chapter 1

From General System Theory to General System Problem Solving

1.1 Introduction

The aim of this chapter is to provide a review that sets the basic theoretical foundations for the thesis. It balances a formalised approach with a more descriptive approach to the necessary basic concepts and terminology. The chapter moves from GST (General System Theory) towards a specific system-theoretic context, GSPS (General System Problem Solving), in which black-box systems can be identified via an inductive reasoning approach based on a pattern-recognition paradigm. Consequently, the paradigm that emerges in this chapter is one of system identification: the behaviour of an unknown system is ascertained from observations and turned into a model. The second half of this chapter elaborates more on GSPS forming the backbone for the tool SAPS (System Approach Problem Solver) described in chapter 2. Throughout the chapter, assumptions with regard to the goal of the thesis are highlighted.

1.2 Systems: the starting point for GST

A system is what is distinguished as a system [Gaines 1979]. The task of scientific research is to find laws describing whole classes of systems and not only unique systems. The resulting discipline is denoted by general systems theory. Klir [1985] expresses this by saying: "*A general system is a standard and interpretation-free system chosen to represent a class of systems equivalent (isomorphic) with respect to some relational aspects that are pragmatically relevant*". Hence, a taxonomy of general systems based on (abstract) relational aspects transcends the disciplines of sciences. This allows a systematic approach later in this thesis where methodological distinctions aid in the taxonomy.

However, concretising the definition is harder than the abstract definition given in the first line of this section. A particular suited definition will be given on the next page. It fulfils the requirements to delineate the approach taken in this work. However, regardless any definition, one still has the separability problem and the selectivity condition that complicates a useful system characterisation [Karplus 1976]. These issues will be recapitulated when talking about a 'source system' in section 1.5.1.

In systems, one is interested, on the one hand in their internal functional relationships, on the other hand in their external relations to the environment. The former is called the structure of the system; the latter is its behaviour [Klír and Valach 1967]. Behaviour is concerned with the dependence of responses to stimuli. Structure is concerned with the manner of arrangement, that is organisation, of the mutual coupling between the elements of the system and the behaviour of these elements.

When observing what are deemed relevant attributes of a system, one must also take into account the space-time specification and the space-time resolution level [Klir 1969].

When a separation of quantities into those produced by the environment and those produced by the system is given in advance, then inputs, the former, and outputs, the latter, can be distinguished. Such systems are called controlled systems [Klir 1969] or directed systems[1], [Klir 1985]. If a separation into inputs and outputs is not given, one talks about neutral systems [Klir 1969].

In this thesis, a first restriction towards directed systems, see Figure 1.1, that represents relatively closed physical systems, is made.

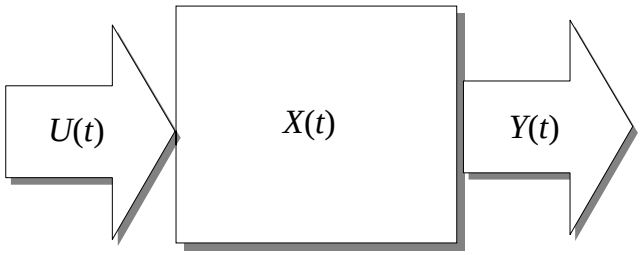


Figure 1.1 : A directed system

For the directed system in Figure 1.1 the external quantities are $U(t)$ and $Y(t)$; the internal quantity is $X(t)$. The input is represented by $U(t)$ and the output by $Y(t)$.

The phrase in [Zeigler 1976], which states that: "*the input-output behaviour of the system constitutes all that can be directly known about a system*" will be the philosophy behind the black-box approach described in the thesis.

There exist several definitions for a system, but they all follow the same underlying principles. In these definitions the concept of states is important to this thesis. This will be explored in more detail in chapter 3. States can store information coming from the inputs if a system has a memory. Klír and Valach [1967] call such systems 'memories'. They also introduced the hierarchical approach that can be found in Zeigler. A definition from Mesarović [1969] and Zeigler [1976] is found in the glossary.

The definition from Klir [1969] is adopted in this thesis. It states that it is sufficient to consider only the following constant traits of a directed system:

(1) A set of external quantities and a given resolution level. In the case of a directed system the external quantities are classified as input and output quantities, and the resolution level is given for both.

(2) A given activity (experimentally determined or observationally given measurements), where the quantities are classified as input and output quantities for a directed system

(3) A given permanent behaviour, which is a time-invariant relation that holds between the principal quantities associated with the latest values of output quantities on the one hand, and the set of all the other principal quantities on the other.

[1] Klir [1969] states that causal systems are directed systems, but the opposite is not necessarily true, while Zeigler [1976] makes no distinction between causal systems and directed systems.

(4) A given UC (Universal Coupling) structure (defined by subsystems, their behaviour, and their coupling). The couplings are directed from output quantities of one element to input quantities of other elements, including the environment for directed systems.

(5) A given ST (State-Transition) structure (defined by a set of states and a set of transitions between them). A more rigorous formulation is given in chapter 3. In the case of directed systems, a single stimulus is associated with each transition.

Depending on the type of the system problem, each of the above traits can be used for a basic definition of a system. They also allow defining 'analysis of systems' and 'synthesis of systems'. The former starts form a given (UC) structure to find the corresponding behaviour and/or the ST structure. The latter starts from a given behaviour or activity and looks for a compatible ST-structure and appropriate (UC) structure. As a particular behaviour can be realised by different structures consisting of the same type of elements, further requirements are needed to determine a structure (see the concept 'inductive bias in chapter 5). The synthesis of systems is far more complicated than the analysis of systems. "*In practice, no case of synthesis can be formulated without the input and output quantities being decided in advance*" [Klir 1969].

1.3 Modelling

A model is a system similar to an original, sometimes called real system in the sense that, when solving a problem concerning the original system, it can solve the problem to a better advantage [Klir 1969]. A model is thus a workable surrogate for a system. Figure 1.2 gives a simplified view on modelling.

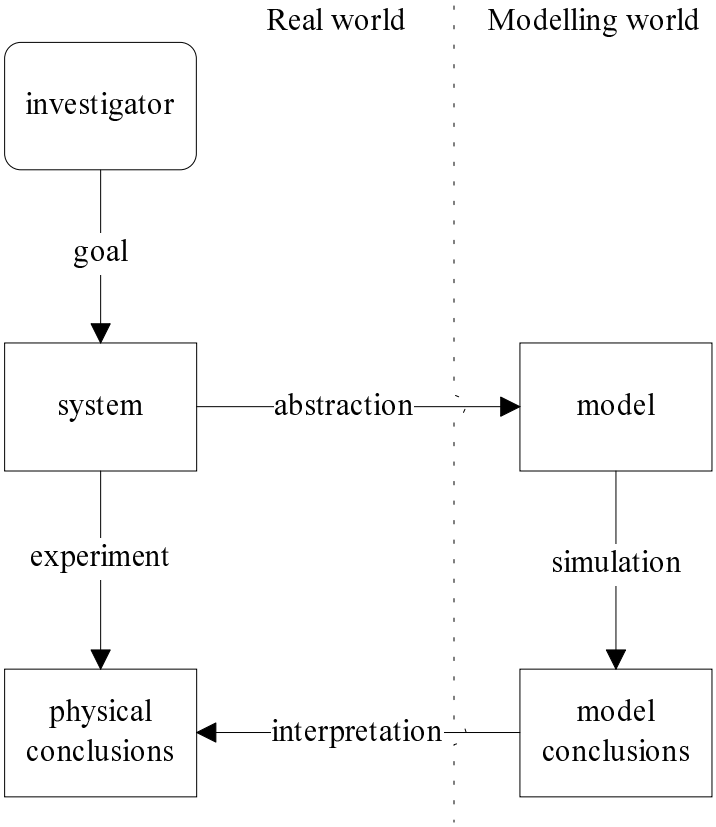


Figure 1.2 : Modelling

Zeigler [1976] considers a base model, but he also has to fall back on a lumped model to have a workable version. The latter is found under a given set of conditions, which is called an experimental frame. An experimental frame embodies the space-time resolution of the system and the purpose of the investigation of the system (e.g., only specifications for real possible behaviour are used). It characterises a limited set of circumstances under which a system is to be observed or experimented with. Hence, it can be defined as "*a definite subset of all possible experiments feasible with a real system and reproduceable with the mechanised model*" [Elzas 1984]. Thus, associated with an experimental frame is a subset of the input-output behaviour of the system under investigation, [Zeigler 1976]. In the terminology of Klir [1985], this corresponds with what he calls an observation channel. Validity of a model is thus always assessed with regard to an experimental frame. For this viewpoint, models and experimental frames are intertwined.

A multidimensional taxonomy for models can be devised, according for example, to their abstraction, to their space-time resolution, to their degree of determinism, their linearity, their emphasis towards structure (e.g. block diagram, Bond graphs, …) or behaviour (e.g. equations). Models are kinds of systems, so they too can be black, white or grey-box. For example, neural networks can be considered black-box, while tree classifiers are white-box (see chapter 8).

In this thesis, a second restriction to white box models for black box systems is made.

1.3.1 Validity of a model: a system – model relation

The validity of a model plays a crucial role in all data experiments undertaken in this thesis. The validity of a model is about how well a model represents the original system it stands for. In the first instance, validity can be measured by the extent of agreement between the original system and the model. This is already formalised in Coombs et al. [1954] where the authors describe two different routes: one is by experiment (validation) and the other by logical argument (deductive modelling)[2]. Both routes try to arrive at the same conclusions about the real world by different means. The notion of validity is extended by Zeigler [1976], who distinguishes different degrees of validity:

1. A model is *replicatively valid* if it matches data already acquired from the original system.

2. A model is *predictively valid* if it can match the data of the original system before these data are acquired from the original system. Predictive validity is stronger than replicative validity.

3. A model is *structurally valid* if it is not only predictively valid, but also reflects the ways in which the original system operates to produce its behaviour.

Philosophically, model building should follow a simple representation of the scientific method in which the need for continued efforts at 'falsifying' the model as an explanation of the observed data is considered as the main methodological activity. Here, the concept of 'conditional validity' (i.e., the model accepted as being conditionally valid until falsified) is introduced in order to allow for inherent uncertainty in life-science systems.

[2] Words in parentheses added by author

1.3.2 Approaches to model building

Some directives should be taken into account when constructing or identifying a model. One has to avoid either excessive complexity or undue simplicity. A model should be representative of the original system and thus reflect the real system closely enough. It should encompass all knowledge available of the original system, otherwise the model may be incomplete [Zeigler 1976]. Yet, it should be parsimonious, i.e., simple enough to understand to achieve the goals of modelling (Occam's razor). Furthermore, complex models are difficult to calibrate and difficult to validate properly. Excessive model complexity can be penalised via the use of complexity measures, e.g. Akaike's measure of fit.

Keeping in mind these directives, two main approaches to model building are devised. They are based on two principal types of information about a general system, which relate to the extreme ends of the system's greyness spectrum (see Figure 1.3):

- knowledge and insight about the system (white box component)
- experimental data of system inputs and outputs (black box component).

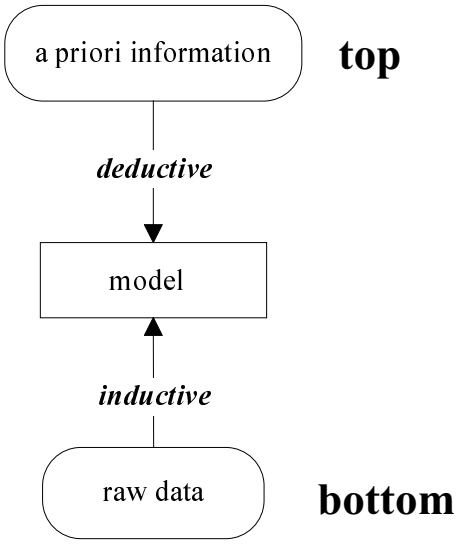


Figure 1.3 : The two sources of information for model construction

The two main approaches are depicted in Table 1.1 and illustrated in Figure 1.3.

	knowledge based approach	data based approach
synonyms	modelling	system identification
	top-down modelling	bottom-up approach
reasoning	deduction	induction
does what?	encodes the (inner) structure of the system	encodes the behaviour of the system (via experimental data)
problem type	Analysis	synthesis

Table 1.1 : The two main approaches to constructing a model

A model constructed purely deductively can generally be considered as a unique solution of modelling. Hence, the top-down approach can and should be used if there is enough a priori knowledge and theory to characterise completely the mathematical equations. In that context, the concept of model structure[3] becomes important.

The bottom-up approach tries to infer the structural information from experimental data and to come up with a usable model under a given experimental frame. This approach may generate an infinite number of models satisfying the observed input-output relationships. So, there is no straightforward procedure for determining the structure of a model. A set of guiding principles and quantitative procedures for inferring structure parts from data sets is needed. More specifically, it is desirable to have additional assumptions or constraints to help selecting an 'optimal' model (this issue will be referred to again when discussing GSPS).

The deductive approach is preferred whenever possible, this is called the physicality principle, because it involves a one-to-one mapping (in fact, no new knowledge is generated), while bottom-up modelling involves a one-to-many mapping (knowledge has to be induced). The latter issue is very important in the field of machine learning. It will be discussed in more detail in chapter 5 (where inductive bias is discussed).

System observations may be obtained either actively or passively. In the former situation the modeller specifies interesting inputs, applies these to the system under study, and observes the outputs. In the latter situation, the modeller cannot specify the inputs and he/she has to accept whatever input-output data is available. Figure 1.4 shows a hierarchy of possible approaches. The inductive approach taken in this thesis is shown in bold.

1.4 Inductive Modelling and System Identification Methodology

If the (deductive) modelling route is impossible, one has to treat the system as a black (or grey) box and try to infer a model via data-analysis of input and output signal recordings. This is the identification route, which is based on experimentation. The Concise Encyclopaedia of Modelling & Simulation describes identification as "*the search for a definition of a model showing the behaviour of a process evolving under given conditions. It is usually realised by means of a set of measurements and by observing the behaviour of a system during a given time interval. The model obtained by identification must enable the evolution of the identified process to be predicted for a given horizon, when the evolution of the inputs and various external influences acting on this system during this time interval are known*" [Atherton and Borne 1992]. Hence, system identification is concerned with the problem of building mathematical models of dynamic systems based on I/O measurements. The approaches to identification as described in the Concise Encyclopaedia of Modelling & Simulation are quite general. They consist of

- collection of input-output data from the system (data is usually recorded by sampling in discrete time),
- settling for a set of candidate models (or model type/paradigm),
- picking one particular member of the model set as the best representative, guided by the information in the data.

[3] Do not confuse with system structure

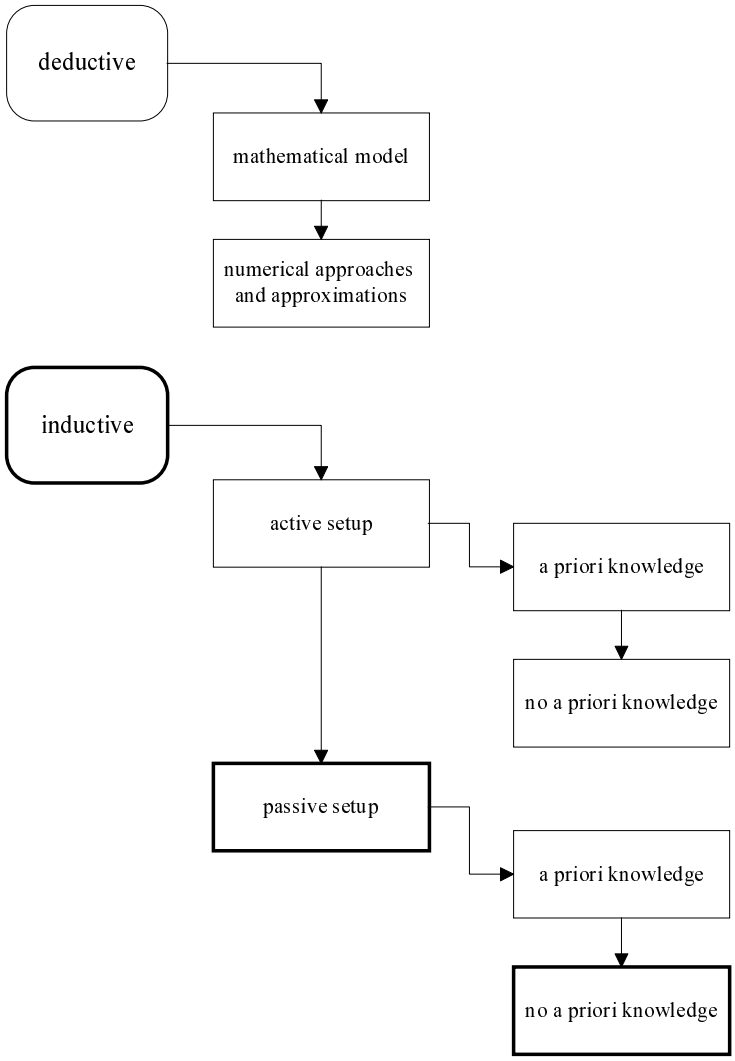


Figure 1.4 : Hierarchy of desirable approaches to modelling

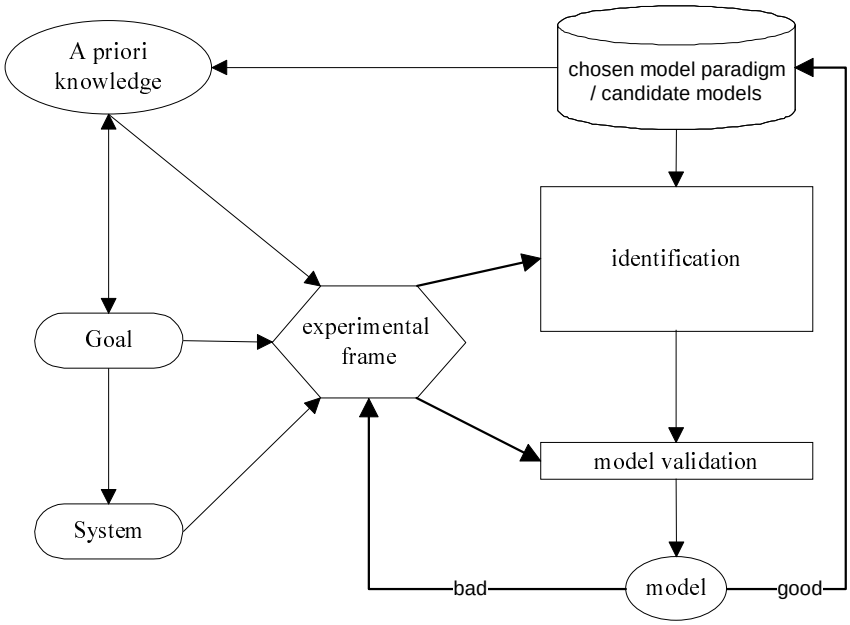


Figure 1.5 : System identification

The identification procedure consists of four choices:

1. Experimental design encompasses the choice of inputs to be made, sampling rates, pre-sampling filters, and so on, to have the most informative data. Experimental design takes into account the goal, the a priori information, and the data. It is mapped into the experimental frame block in Figure 1.5.

2. The choice of a model set or model paradigm is the most difficult one to justify. A priori knowledge and engineering intuition or insight have to be combined with formal properties of model and identification methods to contribute to a good result.

3. The choice of criterion of fit is concerned with the method of evaluating the quality of a particular model. It is put in the experimental frame.

4. The model validation determines if a model is good enough, based on the criterion of fit.

1.4.1 Parametric system identification

In parametric system identification, a model structure is chosen that contains unknown parameters, e.g., one puts forward a certain equation form. This is the structure identification step. Given this structure, the parameters of the structure are identified in a subsequent parameter identification step. Hence, the identification step now consists of two sub-steps: structure identification (compare with choosing a distribution) and parameter identification (compare with filling in the parameters of the distribution). The identification block of Figure 1.5 can now be decomposed as in Figure 1.6. However, if the structure is not right, then all that follows is wrong too.

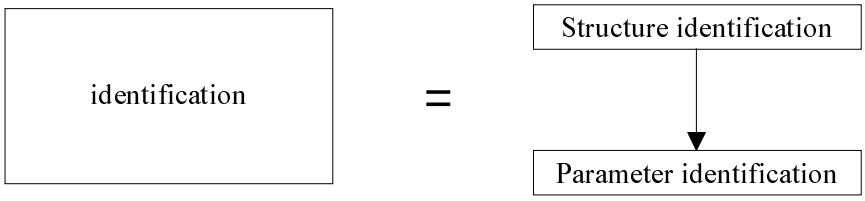


Figure 1.6 : Parametric system identification

A mixed modelling-identification approach in the parametric system identification facilitates a maximum extraction of knowledge from the system. The simplest approach is where the structure identification block is completely replaced by a deductive model construction. Parameter identification then gives estimates for the yet unknown parameter values. The model validation encompass in it also a validation of the given structure: if a good validation can not be obtained, then there is probably something wrong with the given model structure. If one has different model structures available, the parameter identification step should be taken for each candidate model (from the model set): competing models may result. It is hoped that at least the best model from this set passes the validation set.

An example of parametric system identification is found in time series and Box-Jenkins modelling [Ljung 1993]. They work with a certain (linear) model structure that is determined previously. Furthermore, one usually assumes a certain probability distribution for the noise (although research into non-parametric approaches is current). Non-linear parametric system identification is found in the neural network approach [Hecht-Nielsen 1990] and in non-linear time series [Tong 1990]. The next section describes a (non-linear) non-parametric modelling approach.

1.4.2 Non-parametric system identification via general system problem solving

Known non-parametric system identification methods are correlation and spectral analysis, [Ljung 1993]. However, in the approach that forms the backbone of this thesis, the interest is confined to a specific pattern recognition paradigm cast in a theoretical framework, denoted by the name General System Problem Solving (GSPS) [Klir 1985]. It is related to the stepwise, epistemological approach used by humans when doing inductive reasoning in every day situations.

1. They try to identify the problem and determine the inputs relevant for the system.

2. An identification of a subset of characteristics (state variables) related to this problem is made.

3. The manner in which the available inputs influence the related characteristics is determined. This is the search for a pattern.

4. An appropriate strategy to use the available inputs in an optimal manner to achieve the desired characteristics is determined.

GSPS is a conceptual framework through which system problems are formalised in a comprehensive taxonomy of systems. An epistemological hierarchy makes it possible to capture an infinite variety of systems problems through a small number of problem categories. Its underlying principles are based on the notion of external/internal quantities, type of relations, organisation and behaviour of a system.

GSPS starts from the premise that raw data for a given system under investigation are not directly usable for recognising typical system behaviour. They must be converted into a form in which time-invariant relations (more general: time-invariant patterns) between the observed quantities are expressed in a sufficiently simple manner suited to the given purpose [Klir 1969]. The key interest will be the search for a 'good' ('good' will be formalised) time-invariant relation, which is specified by Klir as: "*The time-invariant relation is supposed to be any relation that is satisfied within a certain time interval. In general, some past values and/or some future values of the observed quantities may be included in the relation as well as their instantaneous (present) values*".

1.5 GSPS: a pattern recognition approach to GST

The stepwise inductive approach explored in this thesis is based on the case where a class of systems can be uniquely defined by a given activity (see section 0). It is exemplified by a hierarchy of epistemological levels in which each higher level builds on the lower and adds some additional knowledge. This hierarchy of levels forms the skeleton of the taxonomy of systems in GSPS, see Figure 1.7. Source and data systems are predominantly of an empirical nature, while higher epistemological levels are predominantly of a more theoretical nature.

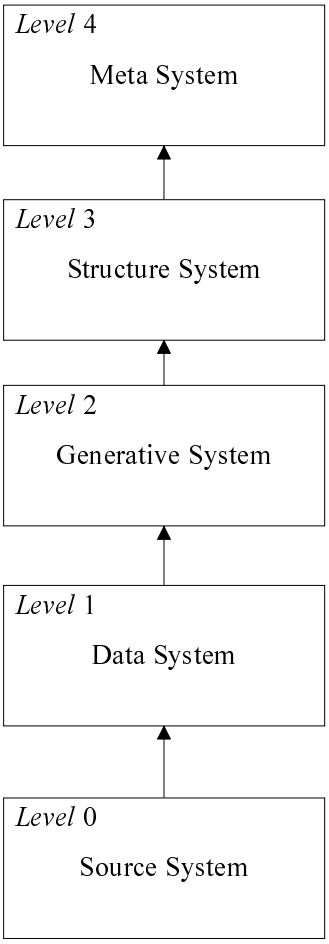


Figure 1.7 : Simplified overview of epistemological level

An inductive approach relates to climbing up the hierarchy. In contrast, a deductive approach signifies descending the hierarchy. As each higher level contains some more knowledge than a lower level, one easily sees that inductive reasoning tries to find more information from facts, while deductive reasoning generates no new knowledge.

At level 0 (source system), one defines what could be the constituents of relevant data relative to the object of investigation, the purpose of the investigation and the constraints posed on the system. At level 1 (data system), one does the actual data gathering, related to the source system previously chosen. The data is stored in an activity matrix. At the next level (level 2), one can start with data processing, where the aim is to discover some support-invariant relationships among the data. Such a relationship will be called a *mask*. Higher levels contain structural and meta-knowledge.

The epistemological levels form the vertical dimension of Figure 1.9. This figure is an elaboration of Figure 1.2. The horizontal dimension indicates which types of system problems are defined (methodological problems or types). Klir considers the methodological distinctions as a secondary classification of system problems. Based on the variables, a first methodological distinction can be made between neutral and directed systems, see Table 1.2.

System type
0 (neutral)
1 (directed)

Table 1.2 : Distinction according to system type

Another — independent or orthogonal — series of methodological distinctions is based on the scale of the variables (and support, see later for a definition of support). They relate to:

- the ordering property : nominal scale or (partial or linear) ordinal scale,
- presence of a metric (distance measures),
- continuity: continuous or discrete.

The different possibilities are shown in Table 1.3.

Ordering	Distance	Continuity
0 (no-ordering)	0 (no)	0 (discrete)
1 (part-ordering)	1 (yes)	1 (continuous)
2 (linear-ordering)		

Table 1.3 : Distinction according to scale

Table 1.3 implies 12 possible combinations. E.g. (1,1,0) depicts a discrete partial ordered variable with a distance measure attached. The combinations (0,0,1) and (0,1,x) are not meaningful, leaving use with nine useful combinations. They can be partially ordered by the relation 'being methodologically more special than' in a Hasse diagram, which is depicted in Figure 1.8.

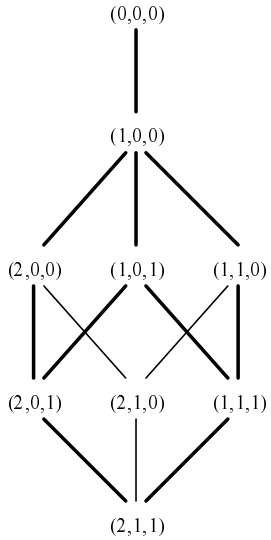


Figure 1.8 : Hasse diagram for methodological types on level 0

Figure 1.9 shows how abstraction leads to a finite number of types of general systems, each characterised by a particular epistemological level and a finite set of relevant and desirable methodological distinctions, [Klir 1985].

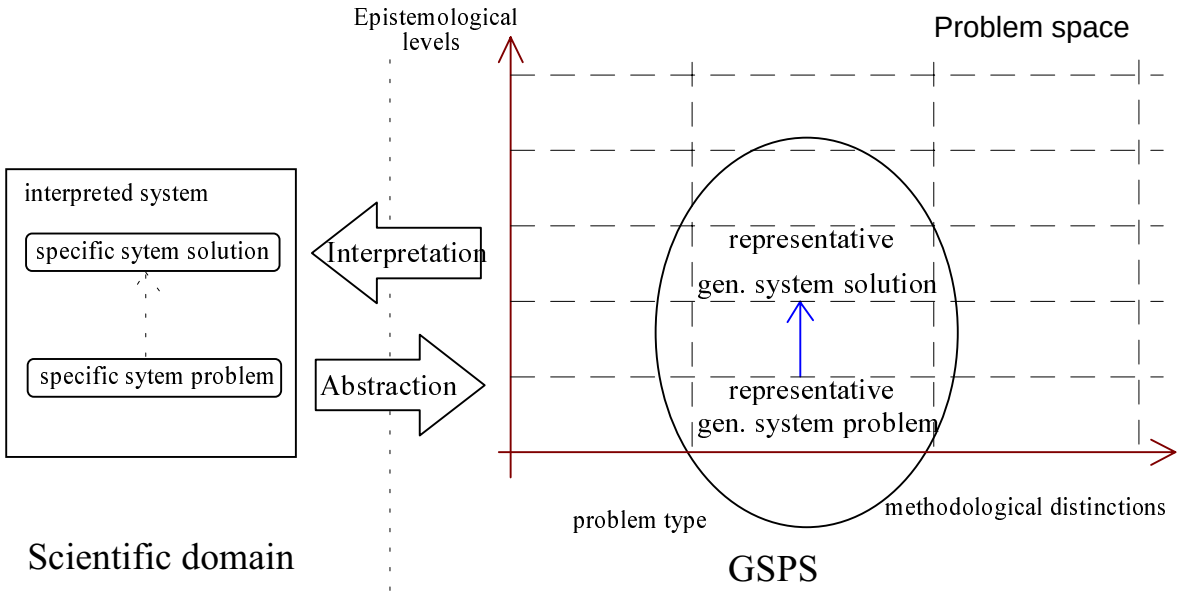


Figure 1.9 : The methodological framework : GSPS

Seeking an ST-structure corresponds to epistemological level 2, while trying to find an UC-structure corresponds to level 3. Only the relevant epistemological levels 1, 2, and 3, needed in the remainder of this thesis are discussed in more depth.

1.5.1 Epistemological level 0: source system (primitive or data-less system)

In general, real world objects consist of a virtually unlimited number of properties, which can not be studied all (at the same time). Remark the relationship with the selectivity problem of Karplus. Therefore, in daily life, interaction of an observer with an object is usually restricted to a limited set of relevant attributes of each object. In order to distinguish individual observations on attributes, an underlying property, denoted by the term ‘backdrop’ (reference variable, index), is needed to distinguish these observations uniquely. A typical backdrop is time; others are space, population individuals and/or combinations of them.

The source system has two main purposes:

- selecting a set of relevant attributes on an object system of interest together with the backdrops (compare with a function of the experimental frame)
- abstracting the relevant attributes and backdrops

This goal is realised via a layered approach in which the source system level is formed by three sub-levels, together with the relations among them. These three primitive sub systems are the object system, the specific image system, and the general image system. They are depicted in Figure 1.10.

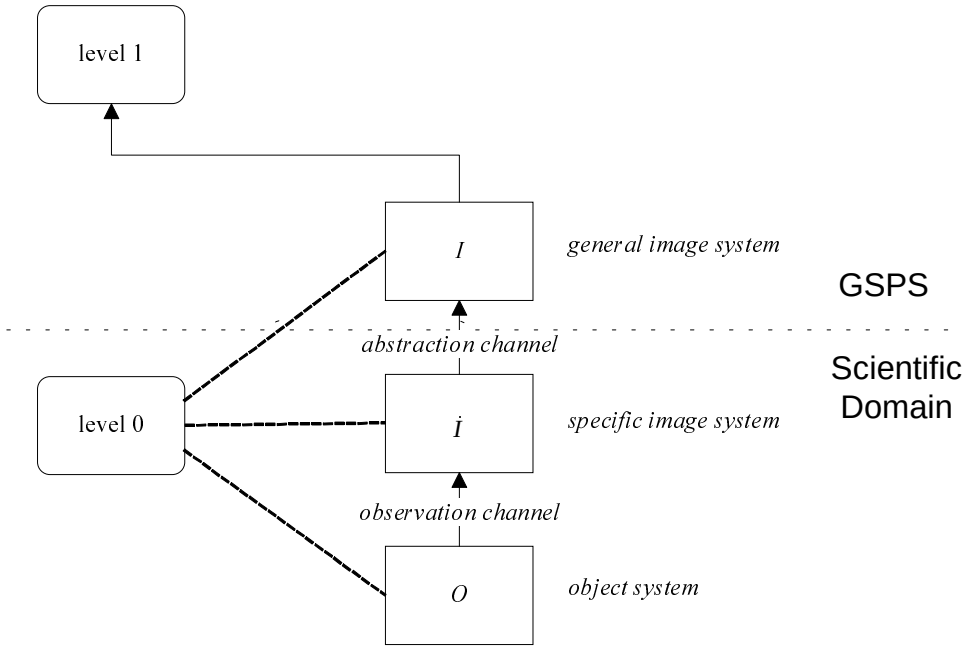


Figure 1.10 : GSPS interface

One notices that the source system interfaces with the real world through observation channels and the object system, and with GSPS through abstraction channels and the general image system. The latter restricts GSPS via abstraction to syntactic (structural and general) aspects of systems problems solving only. The interface with the real world is mediated through the object system *O* and an overall observation channel.

The object system *O*: the 'what' question

Hence, the object system *O* is defined directly on the object of interest by selecting some of its characteristics (attributes and backdrops) relevant to the problem under consideration. An attribute, indexed by *i*, is denoted by a_i and its set of potential appearances by A_i. Variables serve as abstract operational images (representations) of attributes defined in terms of a specific measurement or observation procedure. Each variable, denoted by v_i, has a unique name and is associated with a particular set of entities through which it manifests itself via states or values.

Hence, the object system *O* consists of a set of attributes a_i, each associated with a set of potential appearances A_i, and a set of backdrops b_j, each associated with a set of instances B_j.

Formally:

$$O = \left\langle \{(a_i, A_i) \mid i \in N_n\}, \{(b_j, B_j) \mid j \in N_m\} \right\rangle \qquad (1.1)$$

The choice of attributes (and backdrops) will not only be determined by what one knows, but also by the goal of the identification effort (purpose of investigation). This explains the synonym 'experimental frame' for the object system. What remains is a list of — what are supposed — relevant attributes (and backdrops).

A stepwise approach via a mapping to an image system (first a specific and then a general image system) helps in abstracting the attributes (and backdrops). The term image system was chosen to indicate that the system is an abstract and (usually) simplified representation of some object system. It is a homomorphic image of the corresponding object system.

The specific image system $\dot{I}$: the 'how' question

A first operational abstract (and usually simplified) representation of the object system is defined in an image system. The interaction of the measuring device or observer with the system under investigation is formalised in an observation channel. *Which* characteristic to measure is already determined in the object system. What remains is *how* to measure it.

The specific image system $\dot{I}$ consists of a set of specific variables $\dot{v}_i$ each associated with a specific state set $\dot{V}_i$ and a set of specific supports $\dot{w}_j$, each associated with a specific support set $\dot{W}_j$. Formally:

$$\dot{I} = \left\langle \{(\dot{v}_i, \dot{V}_i) \mid i \in N_n\}, \{(\dot{w}_j, \dot{W}_j) \mid j \in N_m\} \right\rangle \qquad (1.2)$$

The relationship between an object system O and a specific image system $\dot{I}$ is established by an overall observation channel ϑ. It provides a mapping that introduces a specific variable (support) as an image of an attribute (backdrop). During that mapping, some inaccuracies (noises) may be eliminated if not too close to the boundaries. In the latter case, fuzzy techniques may prove useful. A crisp observation channel o_i for an attribute a_i is given by

$$o_i : A_i \rightarrow \dot{V}_i$$

This equation can also be written as

$$o_i : A_i \times \dot{V}_i \rightarrow \{0,1\} \text{ or } o_i : A_i \rightarrow \dot{V}_i \times \{0,1\}$$

As an example, consider the unpopular taxes one each year has to pay. An attribute a_i is the annual income of the person. Its appearance set A_i is {0, 1, ..., 100000} EURO (ignore real millionaires for the moment). The specific variable $\dot{v}_i$ is the annual income bracket the person falls in. The state set $\dot{V}_i$ consists of the tax scale {$x < 3000$, $3000 \leq x < 7000$, $7000 \leq x < 12000$, $12000 \leq x < 20000$, > 20000}. Remark the crisp boundaries on x.

The observation channel is homomorphic with regard to presumed relevant properties of A_i and those in $\dot{V}_i$. It induces a partition in A_i, denoted by A_i/o_i. Elements in each of the equivalence classes are viewed as non-distinguishable through the measurement (observation) procedure. Each induced partition (i.e., the grain size) is called a resolution form, [Klir 1985].

Observations of appearances near the edge of a block can be mapped into the wrong block. These appearances thus have an observation uncertainty, which can be dealt with by the use of fuzzy set theory. In that case, each block is a fuzzy subset of the set A_i or one can consider crisp subsets and assign a degree of certainty that an element belongs to a block. The latter solution is chosen by Klir where a fuzzy observation channel can be seen as a membership grade function that defines a fuzzy relation on the Cartesian product $A_i \times A_i / o_i$. In a more familiar form, this fuzzy relation becomes ($A_i / o_i \rightarrow \dot{V}_i$):

$$\tilde{o}_i : A_i \times \dot{V}_i \rightarrow [0,1] \qquad (1.3)$$

where $\tilde{o}(x, y)$ expresses the degree of certainty that appearance x belongs to the block y in A_i or corresponding value in $\dot{V}_i$. It can be replaced by:

$$\tilde{o}_i : A_i \rightarrow \dot{V}_i \times [0,1]$$

A similar procedure can be followed for the backdrops, but for backdrops, one usually takes the observation points very specific. Thus, mostly no fuzzy set concept is needed[4]. Time is taken as backdrop in this thesis, so there is no need for a fuzzy support-observation channel.

An observation channel can be an explicit definition of the function o_i or ω_j, or operationally when A_i or B_j are not explicitly known. It usually consists of a physical device (measuring instrument) and a procedure describing its use. The (possibly significant) influence of the device on the measured attribute is found in the states of the variable.

Another independent methodological distinction concerns the nature of the overall observation channel: crisp, fuzzy, mixed, see Table 1.4.

Observation channel
0 (crisp)
1 (fuzzy)
2 (mixed)

Table 1.4 : Distinction according to overall observation channel

As already seen from Figure 1.10, observation channels are studied in the traditional sciences and form no part of GSPS itself. An overall observation channel is defined by

$$\vartheta = \left\langle \{(A_i, \dot{V}_i, o_i) | i \in N_n\}, \{(B_j, \dot{W}_j, \omega_j) | j \in N_m\} \right\rangle \qquad (1.4)$$

where

- o_i must be homomorphic with respect to relevant properties in A_i, $\dot{V}_i$
- ω_j must be homomorphic with respect to relevant properties in B_j, $\dot{W}_j$

A fuzzy overall observation channel ϑ could be defined analogously, but with o_i, respectively ω_j replaced by their fuzzy counterparts.

In searching an induced model, it is preferable to abstract the specific variable to facilitate the searching for the model and to standardise the input for higher epistemological levels and the *General* System Problem Solving methodology.

How to deal with noise and measurement errors?

A minimum requirement for an observation channel is that it allows to classify an attribute in an equivalence class in A_i/o_i or, more general, in $\dot{V}_i$. If the blocks of A_i/o_i are substantially larger than the influence of the noise or measurement errors, then the block may correctly be identified with the appearance. However, when the appearance is too close to the edge, the noise (or measurement error) may bring it into a neighbouring interval so that the block is not the right one anymore. The use of a fuzzy observation channel may help. In this case, however, it is proposed to retain the situation where the point is considered to be in both blocks with different membership values. If not, then the point may be classified only in the neigh-

[4] There are cases in which one prefers to use a fuzzy support-observation channel, e.g. consider young, middle-aged and old populations.

bouring block alone (where it should not belong) and the information that it may also belong to the other block (if there were no noise) is ruled out. Chapter 2 will show that the existing tool based on GSPS only retains variables with membership values > 0.5. More research is needed to compare the two options.

The general image system *I*

The GSPS framework only deals with general variables and supports. The necessary extra level of abstraction is achieved by a relationship between a specific image system $\dot{I}$ and a general image system *I* in the form of an overall abstraction - and (in the reverse direction) exemplification channel. Hence, the interface with the higher epistemological levels and GSPS is mediated through the general image[5] system *I*, and an overall exemplification-abstraction channel *E*.

The isomorphic mapping from the specific variables (and supports) to the general variables (and supports) is called an abstraction. Each specific variable (and support) is an interpretation of a general variable (support). The inverse mapping is isomorphic and denoted by the term exemplification. Both mappings are in essence re-labelling functions. They are found in Table 1.5.

	exemplification	abstraction
Variables	$e_i : V_i \to \dot{V}_i$	$e_i^{-1} = \dot{V}_i \to V_i$
Supports	$\varepsilon_j = W_j \to \dot{W}_j$	$\varepsilon_j^{-1} = \dot{W}_j \to W_j$

Table 1.5 : Exemplification and abstraction mapping

As an example, consider again the tax example. The general variable is now simply an integer and the general state set is {1, 2, 3, 4, 5}.

Consequently, a general image system *I* consists of a set of general variables v_i, each associated with a general state set V_i, and a set of general supports w_j, each associated with a general support set W_j, i.e.,

$$I = \left\langle \{(v_i, V_i) | i \in N_n\}, \{(w_j, W_j) | j \in N_m\} \right\rangle$$

An overall abstraction and exemplification channel can be defined in analogy with an overall observation channel.

$$\mathrm{E} = \left\langle \{(\dot{V}_i, V_i, e_i) | i \in N_n\}, \{(\dot{W}_j, W_j, \varepsilon_j) | j \in N_m\} \right\rangle$$

where e_i (ε_j) must be homomorphic with respect to relevant properties in A_i, $\dot{V}_i$ (B_j, $\dot{W}_j$). There is no fuzzy counterpart of *E* for all information is already recoded such that the states set $\dot{V}_i$ is crisp.

The mapping from specific to general image system has as its only means abstraction of data by removing all semantic meaning to facilitate computing. The problem is then generalised, formalised and made domain independent. One may refer to the statement made in [Coombs

[5] The term image system was chosen to indicate that the system is an abstract and (usually) simplified representation of some object system. It is a homomorphic image of the corresponding object system.

et al. 1954]: "*A model is not itself a theory; it is only an available or possible or potential theory until a segment of the real world has been mapped to it*". In this context, one may consider the induction effort as a theory construction via a model creation: the theory construction occurs in the object and specific image system, while the model construction occurs higher up in the epistemological hierarchy. The relation between *O* and *I* is then given by the composition of an observation and abstraction channel.

Summary

The source system can now formally be defined by the quintuple:

$$S = \langle O, \dot{I}, I, \vartheta, E \rangle \tag{1.5}$$

which serves as a *source* for empirical data, and interpretations of abstract data.

Hence, the source system provides a comprehensive frame for data gathering, data processing as well as interpretation of both the data and results of processing.

Any relation recognised in the object system should be made explicit, e.g. ordering, etc. One should aim to get as much a priori knowledge as possible concerning the data expected to be measured (meta-data). Any useful relation that can be recognised in the set of appearances of the attributes, should be added to the definitions of the object system [Uyttenhove 1978], e.g., declarative knowledge should be added too and procedural knowledge must be incorporated. This knowledge must be abstracted in the image system. One should take care not to lose to much information when going to the specific image system. Use the finest resolution if computational complexity stays within limits or use a good quantisation (discretisation) scheme, see the recoding in chapter 2 and the automatic recoding in chapter 8.

A priori knowledge may come from the methodological distinctions (e.g., is the system neutral or directed, is the observation channel crisp or fuzzy, is a variable (or support) nominal, ordered, continuous, ... ?). Derivatives (additional variables) or other derived variables can be added (see chapter 5, augmenting data in the pre-processing step).

The outcome of this effort is a general image system, classified according to its problem type (methodological distinctions) and equipped with all extra knowledge of the system one can get.

1.5.2 Epistemological level 1: data system

The data system contains the information of the source system, but supplemented by data, i.e., by actual states of the variables with respect to a support set. Actual observations are recorded in terms of an ordered pair that consists of an overall support instance at which the observation is made and the observed overall state of the variables involved. A crucial function in the data system is the (crisp) data function, given by

$$d: W \to V$$

where W is the overall support set ($W = W_1 \times W_2 \times \cdots W_m$ for m supports) and V the overall states set ($V = V_1 \times V_2 \times \cdots \times V_n$ for n variables).

While the general image system I characterises only potential states of the variables, function d provides information about their actual states within the delimited support set. Extra knowledge is introduced in the system and the formal definition of the data system is then:

$$D = (I, d) \tag{1.6}$$

For any particular application, however, the meaning of data *d* must be added to the formulation. This is done by replacing *I* with *S* (data system with semantics). For an observation characterised by

$$\underbrace{e^{-1}\Big(\underbrace{o_i(\underbrace{x_i}_{\in A_i})}_{A_i \to \dot{V}_i}\Big)}_{\dot{V}_i \to V_i} = \underbrace{y_i}_{\in V_i}$$

one has $d(w) = v = (y_1, \cdots, y_n)$, where y_i is the observed state of a variable v_i.

The function *d* can be determined in many ways: it may come from observations, it may be derived from higher level systems, or it may be defined in a design.

When one uses fuzzy observation channels, then each actual observation is recorded as an ordered pair that consists of an overall support instance *w* and an *n*-tuple (h_1, ..., h_n) of functions

$$h_i : V_i \to [0,1]$$

where $h_i(y)$ expresses the degree of certainty that *y* is the observed state of variable v_i. In [Klir 1985], data for a crisp observation channel is represented in a matrix

$$d = \left[v_{i,w} \right] \quad i \in N_n, w \in W$$

whose entries $v_{i,w}$ are states of variable v_i observed at overall support instances *w*. This is illustrated in Table 1.6, where the support is specified as discretised time (this will be used later), i.e.,

$$d = \left[v_i(t_j) \right] \quad i \in N_n, t_j \in \mathbb{N}$$

observation point *i*	**0**	**1**	**2**	**…**
v_1	1	2	1	
v_2	2	2	3	
…				

Table 1.6 : A crisp data or activity matrix

Just as an illustration, a fuzzy data matrix is shown in Table 1.7 [6].

[6] The membership values depend on the membership functions. Here, they are mere illustrations and one should not attach too much importance on their values.

observation point *i*	0	1	2	…
v_1	(1;0.9), (2;0.3)	(1;0.5), (2;0.5)	(1;0.7), (2;0.4)	
v_2	(1;0.2), (2;0.8), (3;0.0)	(1;0.0), (2;0.9), (3;0.2)	(1;0.0), (2;0.3), (3;0.7)	
…				

Table 1.7 : A fuzzy data or activity matrix

At the data system level, additional methodological distinctions can be made.

Data can be completely specified if and only if all entries in its data matrix are specified. Otherwise, data is called incompletely specified. In the latter case, two additional types of incompleteness can be specified:

- some data are not available (missing values), see appendix C for an example.
- it does not matter what some of the data are (don't care conditions)

Another methodological distinction can be made when the overall support set is linearly ordered (e.g., when time is the only support). One can then distinguish periodic data from a-periodic data. The examples contained in appendices A, B and D are periodic. The example in appendix C is a-periodic.

The methodological distinctions on level 1 are depicted in Table 1.8.

Spec	Periodic
0 (incomplete)	0 (no)
1 (complete)	1 (yes)

Table 1.8 : Methodological distinctions at level 1

A priori knowledge on the data level can consist of the relevant methodological distinctions. Besides these two distinctions, one should try to obtain statistical information on the raw data (specific image system) and the mapped, sometimes called 'recoded' data (this term will be explained in chapter 2) and do a comparison.

1.5.3 Epistemological level 2: Generative - behavioural system

In the inductive approach, the next step is data processing in which one searches a support-invariant characteristic (a pattern) in the data of level 1. This gives, besides explanation, the possibility for prediction, retrodiction, or generalisation. Prediction will be considered in the examples of the appendices. Support-invariant generative relational characteristics of the variables in the image system are represented by an overall characterisation of a constraint among a set of variables within the support set. In our framework where time is the only support, this boils down to searching time-invariant patterns among the variables.

For an ordered support set, e.g., time, the individual states can be constraint not only by each other, but also by states in a chosen neighbourhood of each particular support instance (pattern depth > 0). Such a neighbourhood is referred to as a mask. A mask is defined in terms of the variables involved, the support set, and a set of translation rules in the support set. A mask

can be represented by a matrix in which the number of rows is equal to the number of variables and the columns relate to a finite set of support instances, see Table 1.9.

v_1	$s_{1,5}$	$s_{2,5}$	
v_2		$s_{3,5}$	
v_3	$s_{4,5}$	$s_{5,5}$	
v_4			$s_{6,5}$

Table 1.9 : A mask with sampling variables for $w = 5$

Each (crisp) translation rule is a one-to-one mapping that accounts for the influence of the neighbourhood obtained via the support set *W*. A translation rule r_j is thus a parameter-invariant relationship given by

$$\begin{aligned} r_j &: W \to W \\ w &\to r_j(w) \end{aligned} \tag{1.7}$$

If the set of transition rules is denoted by *R*, then sampling variables can be characterised by a relationship *M* over *V* x *R*, where *M* stands for the mask defined previously.

$$M \subseteq V \times R$$

Empty entries in the matrix Table 1.9 are not included in the mask (they will be represented by zeros in the masks used from chapter 2 on). Sampling variables are thus non-zero entries in a mask, which can be generalised for fuzzy relations by

$$M \subseteq V \times R \times [0,1]$$

A fuzzy translation rule can be used if the constraint is uncertain. In the sequel, however, a limitation to crisp translation rules is done.

A sub mask is any subset of $V \times R$ for which

$$M_{sub} \subseteq M \text{ or } M_{sub} \in \wp(M)$$

The cardinality of a mask *M*, denoted by #*M*, stands for the number of non-zero entries for the neighbourhood set. The depth of the mask ΔM is defined as the number of columns. The cardinality of *R* is equal to the depth of the mask, i.e., $\#R = \Delta M$, because each particular transition rule *j* acts upon only one column of the mask matrix (see further). The set of sampling variables, denoted by *S*, is then given by $S = \{s_1, s_2, \cdots, s_{\#M}\}$.

Any translation rule r_j can be applied to any variable in v_i in *V*. Thus, given a general image system *I*, a set of variables *V* in *I*, and a set of transition rules, the sampling variables are defined by

$$s_{k,w} = v_{i,r_j(w)} \tag{1.8}$$

Each sample variable can be uniquely labelled by an index function *k*:

$$k : M \to N_{\#M}$$

If s_k is defined solely in terms of a variable v_i (what was supposed), then its state set S_k, is equal to the state set V_i. A special case arises when the support set is *totally* ordered[7], then one can make equation (1.7) more explicit by

$$r_j(w) = w + \rho$$

and thus simplifies equation (1.8), with a more familiar notation for time t, to

$$s_{k,t} = v_{i,t+\rho}, \qquad (1.9)$$

The column in the mask for which $\rho = 0$, is called the reference (instance). Table 1.10 shows a mask. A mask is thus a fixed set of sampling variables.

	ρ = -1	ρ = 0	ρ = 1
v_1	$s_{1,t} = v_{1,t-1}$	$s_{2,t} = v_{1,t}$	
v_2		$s_{3,t} = v_{2,t}$	
v_3	$s_{4,t} = v_{3,t-1}$	$s_{5,t} = v_{3,t}$	
v_4			$s_{6,t} = v_{4,t+1}$

Table 1.10 : A mask with sampling variables for time *t*

Table 1.10 shows the theoretical values of the sampling variables for a reference time t = 5.

v_1	$s_{1,5} = v_{1,4}$	$s_{2,5} = v_{1,5}$	
v_2		$s_{3,5} = v_{2,5}$	
v_3	$s_{4,5} = v_{3,4}$	$s_{5,5} = v_{3,5}$	
v_4			$s_{6,5} = v_{4,6}$

Table 1.11 : A mask with sampling variables for time *t* = 5

where the transition rules can be shown explicitly (#R = 3)

- r_1 (ρ = -1) consists of $s_{1,5} = v_{1,4}$ and $s_{4,5} = v_{3,4}$
- r_2 (ρ = 0) consists of $s_{2,5} = v_{1,5}$, $s_{3,5} = v_{2,5}$ and $s_{5,5} = v_{3,5}$ (identity transition rule)
- r_3 (ρ = 1) consists of $s_{6,5} = v_{4,6}$

It can be seen that a mask placed on an activity matrix only makes a subset of the mask entries transparent, hence, its name. If an activity matrix has k data records, then k-ΔM+1 complete samples of activity can be used from the activity matrix.

The difference between the maximum and the minimum values of r among the sampling elements included in a mask will be called the *memory depth of the mask*, i.e., $\Delta M - 1$. Sampling variables are thus those variables that participate in the time-invariant relations of discrete systems.

[7] Klir already wrote about sampling variables for a totally ordered support in [Klir 1969, p 116]

Different masks can be constructed by using other sets of translation rules. Each mask corresponds to a specific viewpoint according to which the constraint among the variables is represented, [Klir 1985]. A mask can be built up from data or, when it is abstracted from a (more precise qualitative) quantitative description, serve as a means of validation. It is very important to remark that a mask does not change: hence, the translation rules do not change.

The overall state set of the sampling variables C is defined by:

$$C = \prod_{k=1}^{\#M} S_k$$

Data samples are defined as a subset of the overall state set C. They represent observed states of the sampling variables. The subset can be expressed by a functional relationship, called a *behaviour function*. It specifies the occurring states of C, independent of the actual support values. If the behaviour function is crisp, it is called a selection function. The roster method of expressing the constraints is simply to list all possible overall sampling states. This list of samples is a subset of the overall state set of the sampling variables, i.e., the selection function becomes

$$\begin{aligned} f_B: \quad C &\rightarrow \{0,1\} \\ c &\rightarrow 1, \quad \text{if state } c \text{ occurs} \\ c &\rightarrow 0, \quad \text{if state } c \text{ doesn't occur} \end{aligned}$$

The selection function can be considered as a special case of a possibility distribution function, [Klir 1985]. An extension can be found in a fuzzy behaviour function, which is defined by

$$\begin{aligned} \tilde{f}_B: \quad C &\rightarrow [0,1] \\ c &\rightarrow \mu_C(c) \end{aligned}$$

Based on the nature of the behaviour function (crisp, probability, possibility, etc.), different extra methodological distinctions can be introduced. In this thesis, a restriction to probabilistic (special case of a fuzzy measure) behaviour functions is made. This is conform with [Klir 1985] where he considers only possibilistic and probabilistic behaviour functions as most interesting.

Consequently, a behaviour system F_B can be defined to characterise a support-invariant constraint of a set of variables in terms of a behaviour function f_B, a mask M and an image system I.

$$F_B = (I, M, f_B) \qquad (1.10)$$

(each term between brackets is dependent on the previous one(s)).

A mask can be considered as a qualitative representation of the behaviour. Each mask M implicitly represents another aspect (qualitative distinctions; order of system, ...), while each f_B represents the way of selection (crisp, fuzzy). In an implementation of GSPS, called SAPS (System Approach Problem Solver), it will be shown that a probabilistic approach is used. This is already put forward in [Uyttenhove 1978], where he defines a basic behaviour by[8]:

[8] Uyttenhove takes a subset of C, but it can be extended to C by putting zero probabilities for the extra data samples.

$$BasicBehavior = \left\{ (c, p(c)) \,|\, c \in C, 0 \le p(c) \le 1, \sum_{c \in C} p(c) = 1 \right\}$$

Uyttenhove considers also the probability of occurrence of a pair of sample variables. He talks about a basic state-transition relation (ST-behaviour) given by, [Uyttenhove 1978]:

$$BasicST = \left\{ ((c,c'), p(c,c')) | (c,c') \in subset(C \times C), 0 \le p(c,c') \le 1, \sum_{c,c'} p(c,c') = 1 \right\}$$

To be able to generate the data, the basic ST-behaviour is modified by using conditional probabilities instead. Hence, a generative ST-relation is defined:

$$BasicGST = \left\{ ((c,c'), p(c'|c)) | (c,c') \in subset(C \times C), 0 \le p(c'|c) \le 1, \sum_{c'} p(c'|c) = 1 \right\}$$

The constraint, represented by *M*, is a *non-causal* support-invariant relation, which does not describe how data is to be generated. To be able to do that, the sampling variables are partitioned into two distinct sets:

- generating (sampling) variables. They are associated with a generating submask $M_{\bar{g}}$.
- generated (sampling) variables, i.e., they are generated by the generating variables through the support-invariant constraint. They are associated with a generated submask M_g.

A generative mask M_G is then defined by

$$M_G = \left\langle M, M_g, M_{\bar{g}} \right\rangle \tag{1.11}$$

where

$$\begin{aligned} &M_g, M_{\bar{g}} \subset M \\ &M_g \cup M_{\bar{g}} = M \\ &M_g \cap M_{\bar{g}} = \varnothing \end{aligned} \tag{1.12}$$

A re-labelling function can then replace the labelling function *k* by two new ones:

$$\begin{aligned} &\lambda_{\bar{g}} : M_{\bar{g}} \to K_{\bar{g}} \\ &\lambda_g : M_g \to K_g \end{aligned} \tag{1.13}$$

A concrete example is found in chapter 2. The state sets of the

- generating variables are denoted by $\overline{G} = \prod_{k \in K_{\bar{g}}} S_k$
- generated variables are denoted by $G = \prod_{k \in K_g} S_k$

The (crisp) behaviour function now becomes a (crisp) *generative* behaviour function:

$$f_{GB} : \overline{G} \to G$$

The behaviour system becomes a *generative behaviour system*

$$F_{GB} = \left(I, M_G, f_{GB} \right) \tag{1.14}$$

Data can now be generated by a two-step algorithm:

(1) given state $\bar{g} \in \overline{G}$ and $t \in T$, use f_{GB} to determine $g \in G$ for *t*. It is assumed in this case that $\bar{g}$ is known at *t*, so an 'initial' condition is needed.

(2) in- or decrement *t* and repeat (1). Incrementing (for prediction purposes) needs an initial condition for the smallest *t*, while decrementing (for postdiction purposes) needs initial conditions for the greatest *t*. If *t* increases from left to right, then for incrementing, M_g is the set of all right-most elements of *M*, while for decrementing, M_g is the set of all left-most elements.

It is assumed that at least one state *g* is permitted by function f_{GB}, i.e., $\exists g \in G$, $f_{GB}(\bar{g}, g) = 1$. If only one state is permitted, the data generation is unique and the system is deterministic (selection function). If more than one state is permitted, the system is non-deterministic: there is no functional relationship, but a pure relational relationship. The selection function is inadequate for dealing with non-deterministic systems. Consequently, an extension to the popular probability measure is taken. Here, interest lies in the degree of likelihood of occurrence for any state out of the set of all states of the variables involved. One can rewrite the selection function f_{GB} as a relation over $\overline{G} \times G$, i.e.,

$$f_{GB}: \overline{G} \times G \to \{0,1\}$$

$$f_{GB}(\bar{g}, g) = 1 \quad \text{if } g \text{ occurs when } \bar{g} \text{ occurs,}$$

$$= 0 \quad \text{if } g \text{ cannot occur when } \bar{g} \text{ occurs,}$$

then the extension to its fuzzy counterpart is clear via

$$f_{GB}: \overline{G} \times G \to [0,1]$$

$$(\bar{g}, g) \to f_{GB}(g|\bar{g})$$

where $f_{GB}(g|\bar{g})$ is the conditional probability or possibility on $\bar{g}$.

Again, the probability measure will be taken here. If c_g is that portion of a sample *c* that is associated with the generated variables and $c_{\bar{g}}$ stands for the rest of the sample, then a generative relation (generative state transition relation) can be defined as, see [Uyttenhove 1978],

$$GenST = \left\{ ((c_g, c_{\bar{g}}), p(c_g|c_{\bar{g}}))|c_g c_{\bar{g}} \in subset(C \times C), 0 \le p(c_g|c_{\bar{g}}) \le 1, \sum_{c_g} p(c_g|c_{\bar{g}}) = 1 \right\}$$

Selection of a right or optimal mask

The selection of a right mask is governed by general principles as given in [Klir 1985], Hence, the requirements can contain (in decreasing order of importance)

(1) a restriction of a set to a subset by some a priori knowledge, which is by default, or by the user defined, or by expert knowledge, by structural knowledge, etc. (see sub-mask).

(2) a restriction to masks for which the misfit between generated data and experimental data is as small as possible (the selection function can be manipulated accordingly)

(3) a restriction to masks for which the degree of non-determinism is small

(4) a restriction to masks for which the complexity is small

From chapter 2 on, an optimal mask will be looked for that fulfils some requirements. For that aspect, a *largest acceptable mask ML* is defined by $ML = V \times R$ where $R = \{(t+\rho)|\rho_1 \le \rho \le \rho_2\}$ [Klir 1985]. It is a full matrix with n rows and ΔM columns. Each submask M_i represents a restricted set of the possible behaviour systems. Meaningful sub-masks can be derived from a priori (domain) knowledge. At least, they must satisfy the following conditions:

(a) one element in each submask (a row in the *M*-matrix) is included,

(b) one element with the translation rule $t + \rho_2$ must be included (a rightmost element). This requirement is needed to prevent duplicates of equivalent sub-masks.

The number of meaningful sub-masks generated by the domain-independent rules for n variables and depth ΔM is given by

$$N(n,\Delta M) = \left(2^{\Delta M} - 1\right)^n - \left(2^{\Delta M - 1} - 1\right)^n \qquad (1.15)$$

Additional restrictions may be: fixed set of generated sampling variables; fixed number of sampling variables; fixed upper bound of number of sampling variables, or a restriction to masks without gaps (see appendix D for an example of a mask with a gap). These restrictions, together with the background knowledge, result in a set of *candidate masks* (see chapter 2). The background knowledge stems from an amount of a priori knowledge about the system behaviour. E.g. a special rule concerns the fact that for non-ordered support set, one must only take memory-less masks ($\Delta M = 1$). Memory-less masks do not need initial conditions. In contrast, memory-dependent masks require initial conditions. The larger the mask, the larger the number of required initial conditions (order is higher). The set of candidate masks finally resulting from these requirements, should be small enough to let the user determine a suitable mask (if more candidates remain).

To be able, in some cases at least, to determine how well the support-invariant relation is represented by a particular mask, one needs a measure of generative uncertainty associated with the choice of a particular mask. The degree of non-determinism is related to the average uncertainty associated with the generation of data. This uncertainty is defined in terms of generative behaviour functions f_{GB} ($\hat{f}_{GB}$). If the behaviour function is a probability distribution function, then one uses the Shannon Entropy as measure for generative uncertainty. This will be explained in much more detail in chapter 2.

Directed systems

For directed systems the sampling variables can be further partitioned in two subsets:

- the input variables are situated in a submask M_e, the *e* stands for those generated by the environment
- the remaining variables are situated in a submask $M_{\bar{e}}$, hence, $M_e \cup M_{\bar{e}} = M$ and $M_e \cap M_{\bar{e}} = \varnothing$

The mask of a directed behaviour system is then defined by

$$\hat{M} = \langle M, M_e, M_{\bar{e}} \rangle$$

The directed behaviour function f_B is defined by

$$\hat{f}_B : E \times \overline{E} \to [0,1]$$
$$e, \bar{e} \to \hat{f}_B(\bar{e}|e)$$

where the state set of the

- environmental or input variables is denoted by $E = \prod_{k \in K_e} S_k$
- non-input variables is denoted by $\overline{E} = \prod_{k \in K_{\bar{e}}} S_k$

The behaviour system becomes a directed behaviour system

$$\hat{F}_B = \left(I, \hat{M}, \hat{f}_B \right).$$

$M_{\bar{e}}$ can be further partitioned in M_g and $M_{\bar{g}}$ such that (1.12) applies, but now with *M* replaced by $M_{\bar{e}}$. Thus,

$$\hat{M}_G = \langle M, M_e, M_g, M_{\bar{g}} \rangle$$

with

$$\hat{f}_{GB} : E \times \overline{G} \times G \to [0,1]$$
$$(e, \bar{g}, g) \to f_{GB}(g|\bar{g}, e)$$

If the system is deterministic, then it can be rewritten as $\hat{f}_{GB} : E \times \overline{G} \to G$.

Finally, the directed generative behaviour system is defined by

$$\hat{F}_{GB} = \left(\hat{I}, \hat{M}_G, \hat{f}_{GB} \right) \qquad (1.16)$$

The sampling of the data is, if the mask is well chosen, a constant confirmation of the same mask. If this confirmation can not be upheld anymore, one may need to consider the presence of a varying system.

An extended form of behaviour is the state transition relation, which specifies a binary relation of pairs of successive samples. It has some structural connotations [Uyttenhove 1978].

This thesis will only consider probability measures in a directed generative behaviour system. The use of probability measures is more general than simply using a selection function.

Hence, non-deterministic systems, which are more general than deterministic systems, can be dealt with.

1.5.4 Higher epistemological levels

The structure system (level 3) describes relations between lower level systems. The system consists of generative (or lower level) subsystems, which may, for example interact through shared variables. A relation is represented by direct couplings among the elements. It consists of a set of elements and some relations between them.

A meta-system (level 4) shows the relations between lower-level relations. The system consists of lower level subsystems and some support-invariant meta-characterisation (e.g. rule, relation, procedure).

This is a parameter invariant procedure, which describes changes from one system to another.

1.6 Conclusion

This chapter sets the context in which the identification problem described in the introduction is situated. It gives a brief introduction to general systems theory, the two main approaches towards modelling and it gradually focuses from general system theory to a specific non-parametric system identification methodology, called general system problem solving. Some initial assumptions for the framework of this thesis were stated: the systems to be identified are 'black-box' and directed; and the models should be white-box. Furthermore, the observations are obtained passively, which adds a further challenge to the identification process.

References

Atherton and Borne [1992], Concise Encyclopaedia of Modelling & Simulation. ed. D.P. Atherton, P. Borne, Pergamon Press, 1992.

Coombs C.H., Raiffa H., Thrall R.M. [1954], "*Some Views on Mathematical Models and Measurement Theory*", Decision Processes, John Wiley, New York, 1954.

Elzas M.S. [1984], "*System Paradigms as Reality Mappings*", Simulation and Model-Based Methodologies: An Integrative View, ed. Ören T.I., Zeigler B.P. and Elzas M.S., NATO ASI Series, Series F: Computer and System Sciences, vol. 10, Springer Verlag, p. 41-68, 1984.

Gaines B.R. [1979], General Systems research: Quo Vadis? General Systems Yearbook, 24, p. 1-9, 1979. Cited in [Klir 1985].

Hecht-Nielsen R. [1990], Neurocomputing. Addison-Wesley Publishing Company, 1990.

Karplus W.J. [1976], "*The Spectrum of Mathematical Modelling and Systems Simulation*", Simulation of Systems, ed. Dekker L., North-Holland Publishing Company, p. 5-13, 1976.

Klir G.J. [1969], An Approach to General System Theory. Van Nostrand Reinhold, 1969.

Klir G.J. [1985], Architecture of Systems Problem Solving. Plenum Press, 1985.

Klír J. and Valach M. [1967], Cybernetic Modelling. (English edition), Hiffe Books Ltd., 1967.

Ljung L. [1993], System Identification Toolbox - For Use with Matlab. The MathWorks, Inc., 1993.

Mesarović M.D. [1969], "*Mathematical Theory of General Systems*", Advances in Mathematical Systems Theory, ed. Preston C. Hammer, The Pennsylvania State University Press, p. 47-80, 1969.

Tong H. [1990], Non-linear Time Series: A Dynamical Approach. Oxford University Press, 1990.

Uyttenhove H.J. [1978], Computer-Aided Systems Modelling: An Assemblage of Methodological Tools for Systems Problem Solving. Ph.D. thesis, School of Advanced Technology. State University of New York at Binghamton, 1978.

Zeigler B.P. [1976], Theory of Modelling and Simulation. John Wiley & Sons, 1976.

Chapter 2

System Approach Problem Solver

2.1 Introduction

This chapter introduces a non-parametric system identification tool based on GSPS, which is called SAPS (System Approach Problem Solver). Two versions have been designed: one by Uyttenhove [1978] and one by Cellier [1991]. The latter is called SAPS-II. It has more potential for practical applications and so will serve as a reference for the research developed in this thesis. Consequently, this chapter describes SAPS-II in much more detail.

Although SAPS-II is based on GSPS, some methodological differences and restrictions apply. On the source system level, the system type is directed, the support is time, and the observation channel is fuzzy or crisp. The concept of an observation channel can be found in what is called 'recoding' in SAPS-II. At the data system level, the data may or may not be completely specified, [Nebot 1994], but data should be periodic, (see appendix C). At the behavioural system level, only causal masks are used. The matrix representing a mask is transposed with regard to the matrix in GSPS, and is simplified to consider only one output at a time. The latter is no real restriction because, if more outputs occur, then an equal number of corresponding one-output masks is initiated and an identification process is run for each. The evaluation of a mask provides the possibility to compare masks and to search for the best one. Attention is devoted to prediction in SAPS-II for which two methods are available. Both allow the determination of prediction errors [Cellier and Yandell 1987; Herrera 1999]. SAPS-II is also implemented for epistemological levels higher than 2, see for example [de Albornoz 1996], but as in chapter 1 for the theoretical framework, a restriction to the first three epistemological levels is present.

2.2 SAPS

In essence, SAPS uses a pattern recognition approach to identify relevant patterns in data stemming from black box systems by using the general system problem solving framework proposed in chapter 1. As already stated: most identification tools require an assumed model structure to start with (parametric system identification). If the structure is not right, then the parameter estimation that follows is in essence wrong. Thanks to the GSPS framework, SAPS avoids this pitfall by not presupposing any structure (non-parametric system identification). Furthermore, most identification procedures generate a system model that is disassociated from its own confidence information. They allow prediction over a too long time period where the model loses it validity. SAPS draws its own confidence information and does not allow to forecast behaviour that cannot be justified on the basis of observed data (see section 2.3.4).

The first implementation was written in APL by Uyttenhove [1978]. It has been re-implemented as a toolbox under the name SAPS-II within the framework of the CTRL-C environment by Cellier [1987]. Emphasis was put on a cleaner user-interface and a modular approach. A third 'partial' and object-oriented version, called SAPS-ST, is now available as a prototype in Smalltalk, [Van Welden and Vansteenkiste 1994]. More information about this prototype is found in the second part of chapter 4.

2.2.1 The first SAPS implementation

In [Uyttenhove 1978], the author describes a general way in which SAPS should work in the context of GSPS. His implementation is written in APL and concentrates on certain transitions that are considered to be simple because they only imply the application of one kind of methodological tool. Other transitions are compound transitions, which can be formed by combining simple ones. The simple transitions are depicted in Figure 2.1. Each transition problem from epistemological system *i* to epistemological system *j* is denoted by $tr_{i,j}$

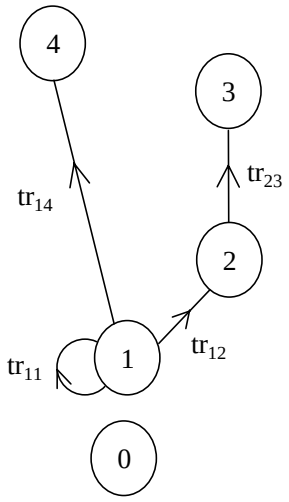


Figure 2.1 : Simple transitions between epistemological levels

The four corresponding tools, one for each simple transition are:

1. Tool AGGREGATE (tr_{11}) deals with transitions on level 1 and allows entering, showing and altering of data. Elementary statistical measures can be obtained about the data. Not only crisp, but also fuzzy data is accepted,. Nine variables and 2000 records can be handled. The mapping from a specific image system to a general image system is performed manually. The transformation from field data to abstract data is called 'recoding': values of a quantity belonging to a discrete system are divided into several disjoint sets (change of resolution). Hence, this tool contains an important routine called RECODE. Thus, recoding, which sets the resolution level of a quantity, in this SAPS version occurs manually.
2. Tool MASKSEARCH (tr_{12}) deals with the mask search (from level 1 to level 2). At this level of behaviour systems, two kinds of behaviours are identified. A basic behaviour is a simple summary of the data in the form of a list of 'aggregate states of the system'. This is a mere collection of individual states observed in the data supplied with their relative frequency of occurrence. A transition behaviour represents the succession of states observed in the system. On this behaviour, a mask has to be fit.
3. Tool LATTICE (tr_{23}) deals with structure constraints (level 2 to level 3).

4. Tool META (tr_{14}) is conceptually defined as a transition through the data system where variation among states of the variables is detected along the dimension of the parameter space (level 1 to level 4).

The concepts accompanying these tools are found back in later implementations of SAPS. Unfortunately, this version of SAPS was not sufficiently flexible to be of much practical use [Cellier 1991]. That is why a second version of SAPS has been developed by Cellier.

2.2.2 SAPS-II

The successor of Uyttenhove's version of SAPS, called SAPS-II, is an interactive software tool coded under the guidance of François Cellier [1987] at the University of Arizona [Cellier and Yandell 1987]. SAPS-II has been implemented as an application function library in FORTRAN to the control systems design software CTRL-C. A version in MATLAB© is also available.

In SAPS-II, major emphasis is put on data flow. The SAPS algorithms are viewed as operators mapping one data structure to another. Each function (or subroutine) is an operator that acts upon incoming data and writes the result of its action via the CTRL-C environment. Another function can be invoked with the resulting data from the previous function. Hence, a good degree of modularity is achieved. SAPS-II furthermore provides a reconstruction analysis based on a established (optimal) mask (level 3) by identifying a structure based on grouping subsets of variables together that seem to be more closely related to each other than to the remaining variables.

Improvements in SAPS-II were done on the determination of the sampling interval. Although there is no precise way to determine the most effective sampling interval, a good rule of thumb is that the mask should cover the dynamics of the slowest mode in the model.

Recoding is refined and extended in SAPS-II. It is argued that recoding can be done more efficiently when it tries to retain as much information as possible. Remarking that the number of states, or levels, that each variable can have after quantisation, is problem dependent, one may state that the number of states should be as low as possible without losing too much information. Hence, the term expressiveness emerged in SAPS-II. It is important to notice that the conflict between the demands of simplicity for the purpose of a strong forecasting power (predictiveness), and an improved resolution for the purpose of a strong expressiveness plays an important role in the choice of the number of levels. The choice of how to map to a chosen number of levels is as in Uyttenhove's implementation fuzzy or crisp, but can now be chosen to be uniform or not. Some quality measures are added and a valid range of forecasting is justified via confidence intervals.

2.2.3 Other research done on SAPS

A research group in Barcelona under the guidance of Prof. Cellier and Prof. Huber has done some valuable research on SAPS. The main emphasis of Nebot's [1994] thesis was to enhance the FIR (Fuzzy Inductive Reasoning[1]) methodology to be able to apply it to soft science systems in general and to biomedical science systems in particular. To this aim she refined the FIR methodology. She also introduced an enhancement of the FIR methodology for dealing with incomplete data records. The thesis of Mugica [1995] treated the systematic design of multivariable fuzzy controllers using FIR. The thesis of de Albornoz [1996] describes a qualitative approach to research in the field of automated plant supervision, fault detection,

[1] FIR is a slight modification of SAPS-II

and fault diagnosis of industrial processes. He has put more effort on reconstruction analysis, which situates itself on the structure system level. The thesis of Herrera [1999] evaluates and enhances the methodology of FIR, such that it helps in the prediction of the future behaviour of time series. Her study allowed a characterisation of the types of time series that FIR predicts well. In order to overcome the ambiguity of the predictions (due to the qualitative approach) new elements of prediction were introduced: the formula used for calculating the relative distances and the absolute weights of the five nearest neighbours was modified, and new confidence measures (based on similarity and proximity) were incorporated. They allow an estimation of the prediction error without the necessity of knowing the true value of the series.

2.3 SAPS in more detail

SAPS concentrates on the observation of a distinct set of quantities at a given space-time resolution, on the search for a simple expression of the time-invariant relation between these quantities, and on the search for the properties that determine the relations mentioned.

Confronted with an (partially) unknown system, one should be able to 'learn' its behavioural pattern via the observation of extracted data. The distribution in the data values is of ultimate importance for if there is not enough variance in these values, one cannot expect to extract useful[2] patterns from it: the data should be representative for the system's behaviour.

2.3.1 Sampling and recoding in SAPS

When gathering raw data for a system, sampling takes place (discretisation[3] of time), [Åström and Wittenmark 1997]. In all existing versions of SAPS the sampled values have to be quantised. Cellier [1991] calls the sampling and quantisation process recoding. In this thesis, recoding is used in a stricter sense so that it only refers to the quantisation process. A fuzzy recoding corresponds then with a composition of a fuzzy observation channel (equation (1.3)) and a crisp abstraction channel (see Table 1.5) for variables, i.e.,

$$e_i^{-1} \circ \tilde{o}_i(x_i) \equiv e_i^{-1}[\tilde{o}_i(x_i)]$$

The sampling process will be discussed first and then the quantisation process.

For the moment, suppose that time is continuous and sampling still has to occur. Usually, this will often be the order in which data is processed as it comes from the system under investigation. The two methods will be described independently.

Cellier [1991] explains how an optimal time step can be chosen according to different principles. He argues that if the time step is too small, the variable has not significant changed and each time instant one is looking at a small time interval for the discovery of the pattern (oversampled situation). On the contrary, if the time step is too big, relevant patterns may be skipped in the step. In this case, with every step the system reaches a new kind of steady state where the steady state values of the state variables depend on the physical input only (undersampled situation).

By considering the system as a n^{th} order system, one may identify a fastest time constant and a slowest one. In order to capture the fastest time constant one has to choose the time step suffi-

---

[2] Useful is used in its general meaning, but in chapter 5 this term will be formalised.

[3] In this chapter, the term discretisation will stand for discretising time (conform digital signal processing terminology). It is not to be confused with quantisation.

ciently small. This can be done via the sampling theorem by Whittaker, which stems from information theory. This states that the time step δt should be (at least twice) smaller than the smallest period observed in the signal. In the examples used in this thesis, data is either already sampled (hopefully fast enough), either generated via simulation in such a way that it is sampled fast enough to capture the fasted time constant one wants to observe. It is hoped that the time step was small enough not to throw away valuable information. It is, however, a different story for the slowest time constant. It has to be captured in the time-invariant pattern that comes into play at the behavioural level. The slowest time constant cannot be discovered always a priori. Therefore, when taking small time steps, masks have to be deep enough (ΔM large).

If a signal has a continuous scale, a way in which to *recode* it can be chosen. This is typically done by first determining the number of desired levels and then to use a quantisation algorithm. A similar process can be used for discrete data points if the level of detail is fine enough. In that case one may still recode the data and map it to smaller value set (amalgamation of levels).

Recoding results in a new qualitative data set of observations. The latter may be called a qualitative activity matrix or a recoded data matrix. It forms the starting point for the next epistemological level.

The recoding process happens in three distinct steps.

First, one determines the number of levels for each variable. A low overall number of levels tends to give good forecasting possibilities but a low expressiveness, while a high number increases the expressiveness but decreases the forecasting power. For example, recoding everything in one level would give a good but useless forecasting power. On the contrary, quantising everything in 1000 levels would give good expressive power, but low predictive power if there were not an enormous amount of points. The trade-off lies in the required minimum number of observations for each composite (overall) level and the variety in observed data level combinations. Cellier [1991] deduced a way to compute the number of desired levels. To explain his method, suppose one has n variables and n_{rec} data records. Each variable is recoded in k_i levels, where a level denotes a value of a discrete state variable. The total number of distinct legal states is the cross product of all possible legal states:

$$n_{leg} = \prod_i k_i \qquad (2.1)$$

According to statistical considerations, one should record each possible discrete state at least five times so the recorded number of states should be at least five times n_{leg}, [Law and Kelton 1990]. In the special case where each variable is recoded in the same number of levels, say k, one obtains:

$$n_{rec} \geq 5k^n$$

Therefore, the preferred maximum number of levels is then

$$k = ROUND\left(\frac{n_{rec}}{5}\right)^{\frac{1}{n}} \qquad (2.2)$$

For reasons of symmetry, an odd number of levels is preferred over an even number. This allows to include a medium range or a zero range as in (-, 0, +). Practically, one does not take the number of allowed levels according to the equation (2.2). The trade-off number of levels, taking into account expressiveness and predictiveness, is 3 or 5 levels (rule of thumb obtained from experience).

The next thing to do is to determine the landmark values. Landmarks are the separation points between the chosen intervals. One may opt for the equidistant case where the landmarks are generating equal intervals of length (fixed-sized recoding) or one can chose the uniform distributed case where the landmarks are put in such a way that the qualitative data distribution is uniform over the intervals (uniform recoding[4]). The way in which data is quantised is very important, because the pattern matching that follows later on, is based on the quantised values. With regard to this aspect, Cellier noticed that certain ways of quantising retain more 'expressiveness' than others. Figure 2.2 and Figure 2.3 illustrate this. The former shows a quantisation with fixed-sized recoding (3 intervals), while the latter shows a recoding (also 3 intervals) under uniform recoding. The latter quantisation retains more expressiveness with regard to fluctuations in the data trajectory. This method is also better because the qualitative values are better backed-up by more real values belonging to them. In fixed-sized recoding it was possible that there were qualitative values representing only a few real values and other representing almost all other real values. Hence, uniform recoding is preferred in this case. However, some a priori considerations may indicate the use of fixed-sized recoding if the separation points between the intervals (landmarks) have a special meaning. This is much related to the case where a domain expert already had an interpretation of the data and where he classified it according to certain criteria. One may then find qualitative terms in the data set.

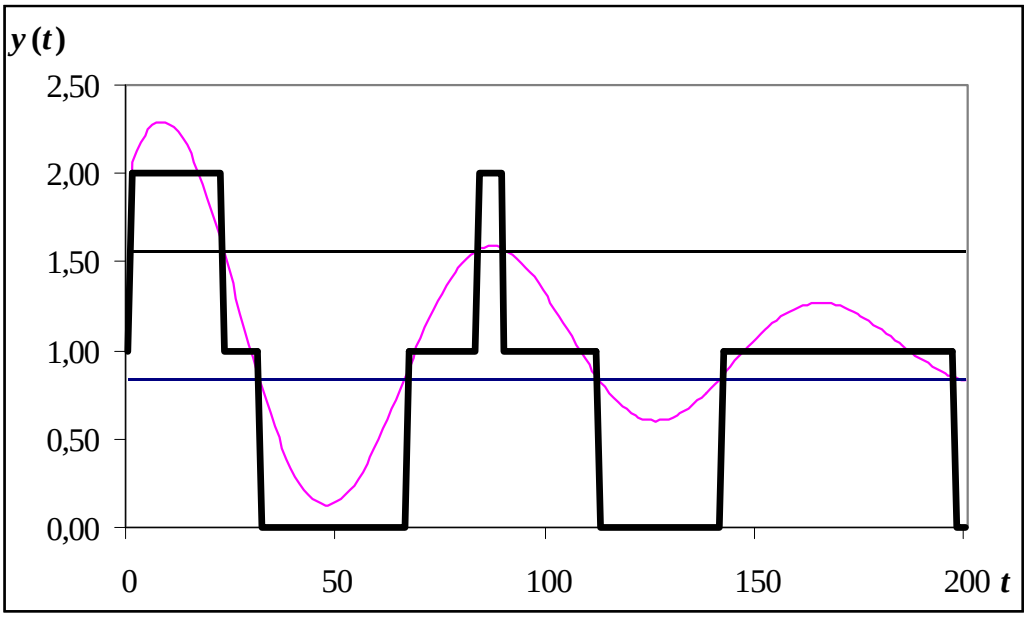


Figure 2.2 : Quantising with fixed intervals

[4] Uniform recoding is equal to an approach based on maximisation of the Shannon Entropy over the given number of intervals.

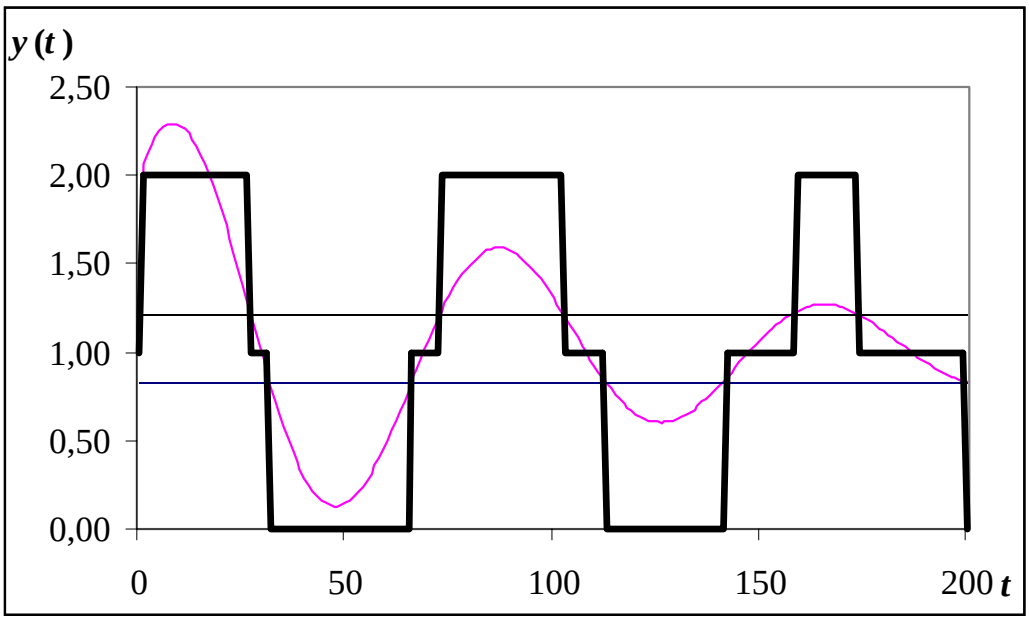


Figure 2.3 : Quantising under a uniform distribution

Finally, when the landmarks are determined in one or another way, one still has two possibilities for continuing the recoding process. Either one maps with a crisp function, or one uses a fuzzy mapping. In a crisp mapping one simply puts the corresponding qualitative value in the qualitative matrix. In the fuzzy case, one uses a membership function to map. This process comes down to fuzzification. Instead of a probability coefficient, the qualitative values get a membership grade coefficient (MGC). The MGC allows calculating the possibility, the confidence and other measures. Its value denotes the degree of confidence one has in the class value for that particular data point. The use of Gaussian membership functions is depicted in Figure 2.4.

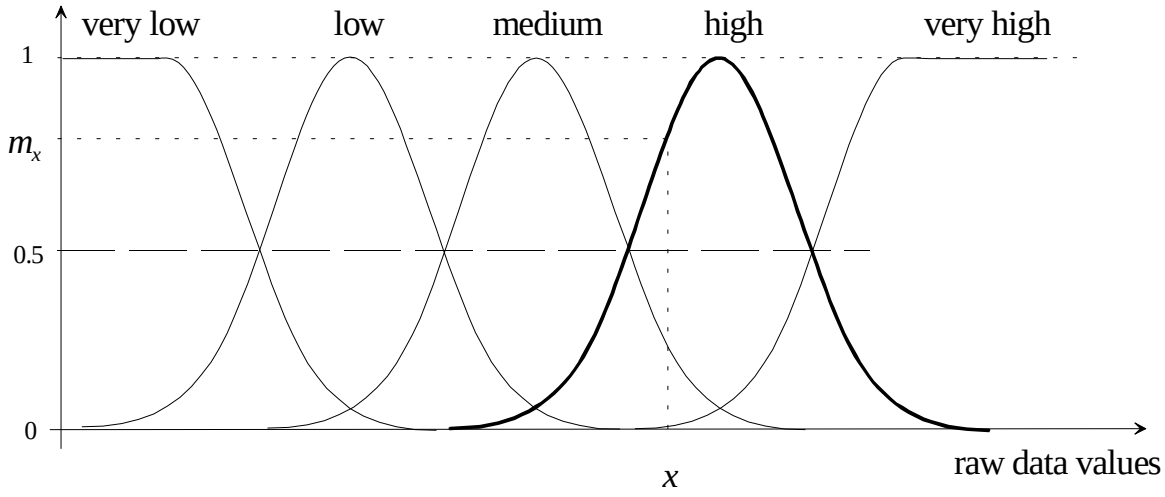


Figure 2.4 : Gaussian membership function

The Gaussian membership function for a continuous variable x is defined by:

$$memb_i = \exp\left(-k_i (x - \mu_i)^2\right) \qquad (2.3)$$

such that $memb_i$ at the landmarks are 0.5 (μ_i is the top or mean of the function).

SAPS-II uses its own kind of fuzzification method. The raw data values are recoded into qualitative triples. The first argument is the class value of the output, while the other two are the fuzzy membership value and the side value. The latter is necessary because of the non-monoticity of the membership function, i.e., left (-1) or right (+1) relative to the top. The side value is something typically for SAPS. This can be seen from Figure 2.4 for a raw (quantitative) data point *x*. Recoding for that point gives the triple (high, m_x, -1). Hence, recoding a raw data matrix of the form

t	u_1	u_2	u_3	u_4	*y*
1	3.5	2.4	5.0	3.0	10.2
2	7.2	8.7	2.1	1.4	8.3
3	…	…	…	…	…

Table 2.1 : Raw activity matrix

gives rise to three new matrices after fuzzy recoding[5], i.e., the matrix with the set values (Table 2.2), the matrix with the membership values (Table 2.3) and the matrix with the side values (Table 2.4).

t	u_1	u_2	u_3	u_4	*y*
1	low	very low	medium	low	very high
2	high	very high	very low	very low	high
3	…	…	…	…	…

Table 2.2 : Set values after recoding Table 2.1

t	u_1	u_2	u_3	u_4	*y*
1	0.80	0.55	0.81	0.79	0.67
2	0.72	0.64	0.92	0.95	0.69
3	…	…	…	…	…

Table 2.3 : Membership values after recoding Table 2.1

t	u_1	u_2	u_3	u_4	*y*
1	-1	-1	-1	+1	+1
2	-1	+1	+1	+1	-1
3	…	…	…	…	…

Table 2.4 : Side values after recoding Table 2.1

[5] The numbers are merely for illustration purposes.

These together refer to a qualitative data model, [de Albornoz 1996].

The I/O variables are now recoded to a fixed set of discrete values (categorical variables). These are stored in a data matrix, which serves as primary input for the pattern identification process. SAPS will try to find a pattern that explains the data present in the set matrix, depicted in Table 2.2. The identification of a time-invariant pattern occurs in two phases:

1. Apply a mask, evaluate it somehow so that comparisons are possible
2. Search if there is a better mask with regard to the chosen evaluation criterion

These two steps will now be described in more detail in section 2.3.2 and section 2.3.3.

2.3.2 Evaluation of a time-invariant pattern or mask

Basically, there are two important patterns for the model identification in SAPS. The primary pattern is what is already defined as a mask in GSPS. It represents the input-output dependency relations induced from the system. The secondary pattern is the state-transition matrix describing how the relations are spelled out in a discretised form. It is needed to define what is called the quality of the corresponding mask.

In SAPS, the primary patterns or masks are cast in matrix notations where the time axis (contrary to GSPS where the matrix is in a transposed form) now stretches over the rows of the matrix. An example for three input variables (u_1, u_2, u_3) and one output variable (y) is given below. Just for convenience the output is put as the last column.

	u_1	u_2	u_3	y
$t-3\delta.t$	−1	0	−2	0
$t-2\delta.t$	0	0	0	0
$t-\delta.t$	0	−3	0	0
t	0	0	−4	1

Table 2.5 : A mask in SAPS

Although for directed systems three kinds of entries, (input, generating and generated, see chapter 1) can be identified, it is sufficient in this context to distinguish only between generating and generated entries, called *m*-inputs and *m*-outputs[6], [Cellier et al. 1996]. The former are denoted by negative numbers, the latter by positive numbers. The different negative numbers are just to specify the ordering when flattening the recoded data. This is the concretisation of the labelling functions shown in equation (1.13), where

$$K_{\bar{g}} = \{-1, -2, -3, \cdots\}$$
$$K_{g} = \{1, 2, 3, \cdots\}$$

I.e., generating sampling variables are represented by negative numbers, while generating sampling variables are represented by positive numbers.

[6] In SAPS-II, no further distinction in generated, generating, and input sampling variables is done for practical reasons.

Table 2.5 represents a mask with *depth* four (ΔM = 4). Alternatively, one could say that the mask has a memory of three time steps, thus a *memory depth* of three. The mask in Table 2.5 represents the following relationship:

$$y(t) = \tilde{f}(u_1(t-3\delta.t), u_3(t-3\delta.t), u_2(t-\delta.t), u_3(t)) \qquad (2.4)$$

where $\tilde{f}$ stands for a qualitative relationship.

Shifting a mask over the recoded data results in an Input/Output matrix, which shows in each row the sampling state. Compare this with equation (1.9) and Table 1.10 from chapter 1. Chapter 3 gives an example in another theoretical context.

t	$s_{1,t} = u_1(t\text{-}3\delta t)$	$s_{2,t} = u_3(t\text{-}3\delta t)$	$s_{3,t} = u_2(t\text{-}\delta t)$	$s_{4,t} = u_3(t)$	$s_{5,t} = y(t)$

Table 2.6 : An Input/Output matrix row

The secondary patterns are the state-transition matrices. They are created by moving a mask over the recoded data and by summarising the set of sampling variable values. A more detailed description of this process will be given in chapter 3. At this moment, it is sufficient to realise that with each mask there belongs a corresponding state-transition matrix. An example of a state-transition matrix is found in Table 2.7 (this state-transition matrix cannot be derived from the mask in Table 2.5, because the dimension of the overall generating state set is different).

in \ *out*	1	2	3	4	p_{in}
11	0	0.667	0.333	0	0.3
12	0.333	0	0	0.667	0.3
21	0	0	1	0	0.2
22	0.5	0.5	0	0	0.2

Table 2.7 : A state-transition matrix in SAPS

The state-transition matrix contains the input-output relations and forms the backbone for the forecasting abilities of SAPS. The leftmost column of Table 2.7 contains the observed overall generating states (*m*-inputs), while the top row shows the possible values of the generated sampling variable[7] (*m*-output). The entries in the matrix (second to one-but-last) column contain the conditional probabilities of occurrence (see further). The last column contains the estimated probabilities of occurrence of the observed generating states.

The state-transition matrix can be translated to rules, which then can be used in an expert system. In this case, the expert system may learn from unknown behaviour and store the extracted and transformed state-transition matrix in its knowledge base. SAPS can thus also be seen as an addendum to expert systems. It enhanced the possibilities of the latter. In chapter 8 another way of generating rules via another paradigm is shown.

The complexity of the state-transition matrix depends on the cardinality of the mask. One could use the data matrix itself as a (secondary) pattern, but this pattern is regarded as too complex and unusable for forecasting (parsimony principle for modelling). Hence, more simple patterns have to be found by first searching for simple (lower cardinality of mask) input-

[7] As only one generated variable per mask is taken, just refer to it as the output.

output relations over time. Thus, a simple mask has to be found. However, it cannot be too simple, for then it would generate too much uncertainty in the prediction. A trade-off has to be found.

The trade-off will be represented in an evaluation function, denoted by *Q*. In SAPS terminology, this is called the quality of a mask (although it also stands in fact for the quality of the state-transition matrix). *Q* is a function of relevant measures on the primary and corresponding secondary pattern. Three basic kinds of measures will come into play:

- the degree of determinism (or uncertainty) of the candidate model (based on the secondary pattern),
- the complexity of the candidate model (based on the primary or secondary pattern),
- the predictiveness of the candidate model (based on the secondary pattern).

Modifications/extensions on this will be discussed in chapter 4.

Determining the entropy of a mask for crisp recoding

The degree of determinism will be computed via the Shannon Entropy. Thus, it is proposed to take the (conditional) Shannon entropy as a measure for the uncertainty allowed in the behaviour prediction. Shannon entropy is a measure for the information-contents. It is defined by

$$H = -\sum_j p_j \log_2 p_j \qquad (2.5)$$

where each outcome *j* (output) has a probability p_j of occurrence. If a certain outcome, say outcome *k*, has a probability 1 of occurrence (and consequently the others have a probability of 0 of occurrence) then the entropy is 0. In this case one has a minimum of uncertainty or a maximum degree of determinism. In case all probabilities are equal (uniform distribution of probability of occurrence) then the entropy will be maximal. For *n* outcomes, each outcome will have a probability of (1/*n*), and thus substituting this in expression (2.5) gives

$$H_{\max} = \log_2 n$$

In this case one has no clue of which outcome is most likely to occur and thus the degree of uncertainty is maximal or the degree of determinism is minimal. Concluding, a small entropy value is desired to reduce the uncertainty.

The use of entropy in SAPS is based on conditional probabilities of the output. For each data row in the state-transition matrix (such as the one in Table 2.7), one can compute the entropy of the outputs (indexed by *j*) conditional on a given *m*-input state[8] by

$$H(out \mid input_i) = -\sum_j P(output_j \mid input_i).\log_2 \left[P(output_j \mid input_i) \right] \qquad (2.6)$$

The total entropy for a given state-transition matrix is then a weighted entropy over the *m*-input states (indexed by *i*), or the average entropy over the *m*-input states:

$$\begin{aligned} H_m &= \sum_i P(input_i).H(out \mid input_i) \\ &= -\sum_i P(input_i).\sum_j P(output_j \mid input_i).\log_2 \left[P(output_j \mid input_i) \right] \end{aligned} \qquad (2.7)$$

[8] The *m* in *m*-input is dropped for convenience.

and this defines the Shannon entropy as used in SAPS. It is the conditional Entropy of the output given the input. Applying this on the state-transition matrix in Table 2.7 gives

$$\begin{aligned} H_m = &-\{0.3*\left[0.667*\log_2(0.667)+0.333*\log_2(0.333)\right] \\ &+0.3*\left[0.333*\log_2(0.333)+0.667*\log_2(0.667)\right]+0.2*\left[1*\log_2 1\right] \\ &+0.2*\left[0.5*\log_2 0.5+0.5*\log_2 0.5\right]\} \\ &=0.75 \end{aligned}$$

As the state-transition matrix is used for forecasting, it is important to have a high degree of determinism and, consequently, to strive for low values of the entropy. However, the model for prediction has to be as simple as possible (under certain restrictions). This is the principle of parsimony. Hence, a complexity measure has to be introduced. The complexity measure can be defined in many ways. It can be based on the number of *m*-inputs, the depth of the mask, etc.

Determining the entropy of a mask for fuzzy recoding

The quality determination in the previous section is only valid for crisp recoding where a measure of 'confidence' is the probability of occurrence of a given *m*-input/*m*-output record. With 'confidence' is meant what Cellier [1991] says: "*it specifies how confident we are that the assigned value is correct* (in a certain *m*-input/*m*-output record)[9]".

In the case of fuzzy recoding, fuzzy membership functions are to be used to obtain a measure of 'confidence'. In Table 2.8 one sees a record from applying a mask to a certain row from a recoded data matrix with the corresponding membership values (side values not shown).

Recoded data or Input/Output matrix		
x_1	x_2	x_3

Membership values		
m_1	m_2	m_3

Table 2.8 : A *m*-input/*m*-output record and its associated membership values

The confidence of a data row *r* is determined by the joint membership of the individual membership values, i.e.,

$$c(input_{i,r}, output_{j,r}) = memb_{\text{joint},r} = \bigcap_{\forall k \text{ in row } r} memb_k = \min_{\forall k}(memb_k)$$

where *k* loops over the *m*-inputs and the output in row *r*.

The rest of the process for constructing a state-transition matrix is pretty much the same as in the crisp case, but the accumulation of confidences may give values larger than 1. This is because SAPS uses a sum for the union of data records with the same state[10], i.e.,

$$\begin{aligned} c(input_i, output_j) &= \bigcup_{\substack{r, \\ (input_{i,r}, output_{j,r}) = (input_i, output_j)}} c(input_{i,r}, output_{j,r}) \\ &= \sum_{\substack{r, \\ (input_{i,r}, output_{j,r}) = (input_i, output_j)}} c(input_{i,r}, output_{j,r}) \end{aligned}$$

[9] Parenthesis added

[10] Other definitions for the union, such as maximum, may be applied.

To be able to use the same entropy function as before, normalisation is mandatory. Although the use of the Shannon entropy in this case is theoretically not entirely correct (as Cellier [1991] remarks), it seems to work satisfactory. There are theoretical solutions for this (the use of U-entropy) foreseen, but they are not implemented (yet). Hence, in SAPS-II the entropy is calculated as in the crisp case, but with the conditional probability replaced by a relative confidence, i.e.,

$$P(out_j / input_i) = \frac{c(input_i, output_j)}{\sum_j c(input_i, output_j)} \tag{2.8}$$

Complexity measures

Just to fix the idea, suppose complexity is defined as the number of *m*-inputs, i.e., #*M*-1. A mask with few *m*-inputs tends to give state-transition matrices that have a low degree of determinism, while masks with many *m*-inputs give rise to state-transition matrices with a high degree of determinism. This corresponds with the principle that simpler models tend to have less expressive power (lower resolution) and thus a higher degree of uncertainty. So, a trade-off between degree of determinism and complexity has to be found.

The evaluation function *Q* should be such that the trade-off is incorporated in its functional description. Thus a first proposal would be to use a function of the form

$$Q = \frac{1 - H_m}{C} \tag{2.9}$$

where the degree of certainty could be defined by $(1 - H_m)$ and where H_m is defined by equation (2.7). This, however, is inappropriate because one needs to compare different state-transition matrices (and their corresponding masks) with each other to find the best model. Hence, normalisation is necessary for comparison among models (i.e., comparing state-transition matrices with different number of *m*-inputs). This is obtained by defining a normalised overall entropy reduction by

$$H_r = 1 - \frac{H_m}{H_{\max}} \tag{2.10}$$

Applied to the previous example, the normalised overall entropy reduction becomes

$$\left.\begin{aligned} H_m &= 0.75 \\ H_{\max} &= \log_2 4 = 2 \end{aligned}\right\} \quad H_r = 1 - \frac{0.75}{2} = 0.625$$

A possible evaluation function is now

$$Q = \frac{H_r}{\#(m - \text{inputs})} = \frac{1 - \frac{H_m}{H_{\max}}}{\#M - 1}$$

for a state-transition matrix with a corresponding one-output mask *M*.

Striving for small *H* is equal to striving for large certainty values (the numerator), and striving for low complexity is striving for a low mask cardinality (the denominator). Thus, in both cases, one aims at a high mask quality.

In a first version of SAPS-II, Cellier and Yandell [1987] used the quality function

$$Q = \frac{H_r}{C} \tag{2.11}$$

where the complexity weight C for an applied mask M is defined as

$$C = \frac{n_{\text{var}}.\Delta M.\#M}{\Delta M_{\max}}$$

where

- n_{var} is the number of variables,
- ΔM is the mask depth,
- $\#M|$ is the number of non-zero entries in the mask.

In a later version of SAPS-II, another measure was introduced, called the observation ratio (OR), [Li and Cellier 1990]. It is a measure for the predictiveness of the prediction model (state-transition matrix). It is defined by

$$OR = \left(\frac{5n_{\geq 5} + 4n_4 + 3n_3 + 2n_2 + n_1}{5n_{leg}} \right) \tag{2.12}$$

where

- n_{leg} = total number of distinct legal input states
- n_1 = number of input states observed only once
- n_2 = number of input states observed only twice
- n_3 = number of input states observed only thrice
- n_4 = number of input states observed only four times
- $n_{\geq 5}$ = number of input states observed five times or more

So if every legal input state has been observed at least five times, then OR = 1. If no data has been observed, then OR = 1. OR will thus lie between 0 and 1.
The new form of the evaluation function (quality of a mask) is now defined by

$$Q = H_r \times OR = \left(1 - \frac{H_m}{H_{\max}} \right) \times OR \tag{2.13}$$

This function gives also a rating of how good (according to our new criterion) a mask performs. The choice of a quality function and the choice of how and which measurable features to include, is quite empirical and subjective. It is obvious that the applied criterion measures must be based on sound principles. Besides the two choices just discussed, others can be devised. In chapter 4, some other definitions are given.

This section showed how to evaluate a mask via its quality. Once a quality function is defined, what remains is how to find the optimum quality value and its corresponding pattern. The corresponding optimal mask guarantees that according to our chosen criteria, an optimal pattern has been found that explains the data.

2.3.3 Searching the best mask

Section 2.3.2 demonstrated how to evaluate a mask and its corresponding state-transition matrix. The latter can serve as a prediction model, see section 2.3.4. However, for a given data set, many masks (and predictive models) are possible. This brings us to the search for the best mask from a set of all possible candidates for a given data set (thus for a given source and data system).

This set can be very large for a reasonable large data set. One could take as set the mask that covers the whole data set, i.e., $\Delta M = n_{rec}$ with all its sub-masks (see equation 1.15). Hence, an upper bound on the complexity has to be inserted by the user in the form of a candidate mask. SAPS ensures it will only try out and evaluate sub-masks from this candidate mask. Section 1.5.3 in chapter 1 shows how to define a sub-mask, but here a less formal description for one-output sub-masks is given. A sub-mask M_1 from a mask M is formed by putting certain m-inputs in M zero. However, finding a relevant mask is far from trivial because of the fact that — in the realistic case — different valid patterns or masks can be found that all explain the observed data records under certain chosen criteria, expressed by a quality function Q. Hence, the candidate mask must be carefully chosen and preferably be based on a priori knowledge.

It has been argued that in order to capture the fastest time constant, one has to choose the time step sufficiently small. This should have been done in the source system. However, the slowest time constant can be captured via the mask. One can do this by simply taking the mask large (deep) enough to hold this time constant. Cellier [1987] demonstrated this issue by simulating a first linear order system with one input and three outputs. So one rule to take into account is the following: *"The depth of the mask must approximately cover the slowest time constant of the system we want to capture"*. Section 2.3.1 showed that the sampling period δt should be no larger than one half of the shortest time constant. Using such a sampling period, the slowest time constant should be captured by the time span covered by the mask, which is denoted by Δt. Each additional row in a mask adds an extra δt to its total time span, thus

$$\delta t.(\Delta M - 1) \geq \Delta t$$

From this equation, the advisable mask depth can easily be computed.

Obviously, one could try to take the mask very deep (pessimistic vision) to be sure to have captured the slowest time constant. However, an upper bound on mask depth is needed to restrict the computation load to a manageable size.

Every possible sub-mask of the candidate mask can be considered as a point in a search space delimited by the initial candidate mask. In SAPS-II, one does an optimal mask search in this space, which implies an exhaustive search among all sub-masks in the bounded search space.

For a given output variable (a one output-mask), one starts from the empty variable set (the initial state). All operators, each of which adds a variable, are applied to it so that all two entry-states (one m-input and one m-output) are generated, i.e., the cardinality of these masks is two. All corresponding sub-masks are evaluated by the same quality function. The one with the highest quality is retained. Then one creates all masks with an extra m-input entry and starts evaluating all states formed by combinations of these m-input entries. Again, the one with the highest quality is retained. This process continues for higher entry masks until the candidate mask is reached. The mask (state) with the absolute best quality is then chosen to be the optimal one (goal state). An improvement by some techniques to stop excessive evaluation can be built in by cutting the generation of new masks under certain observations. Apart from this improvement, one notices that the search is exhaustive. The *optimal* mask will be

found (within the constraints of a candidate mask). All the others will have lesser qualities. The search process is graphically depicted in Figure 2.5. Remark the 'bottom-up' approach in the search process.

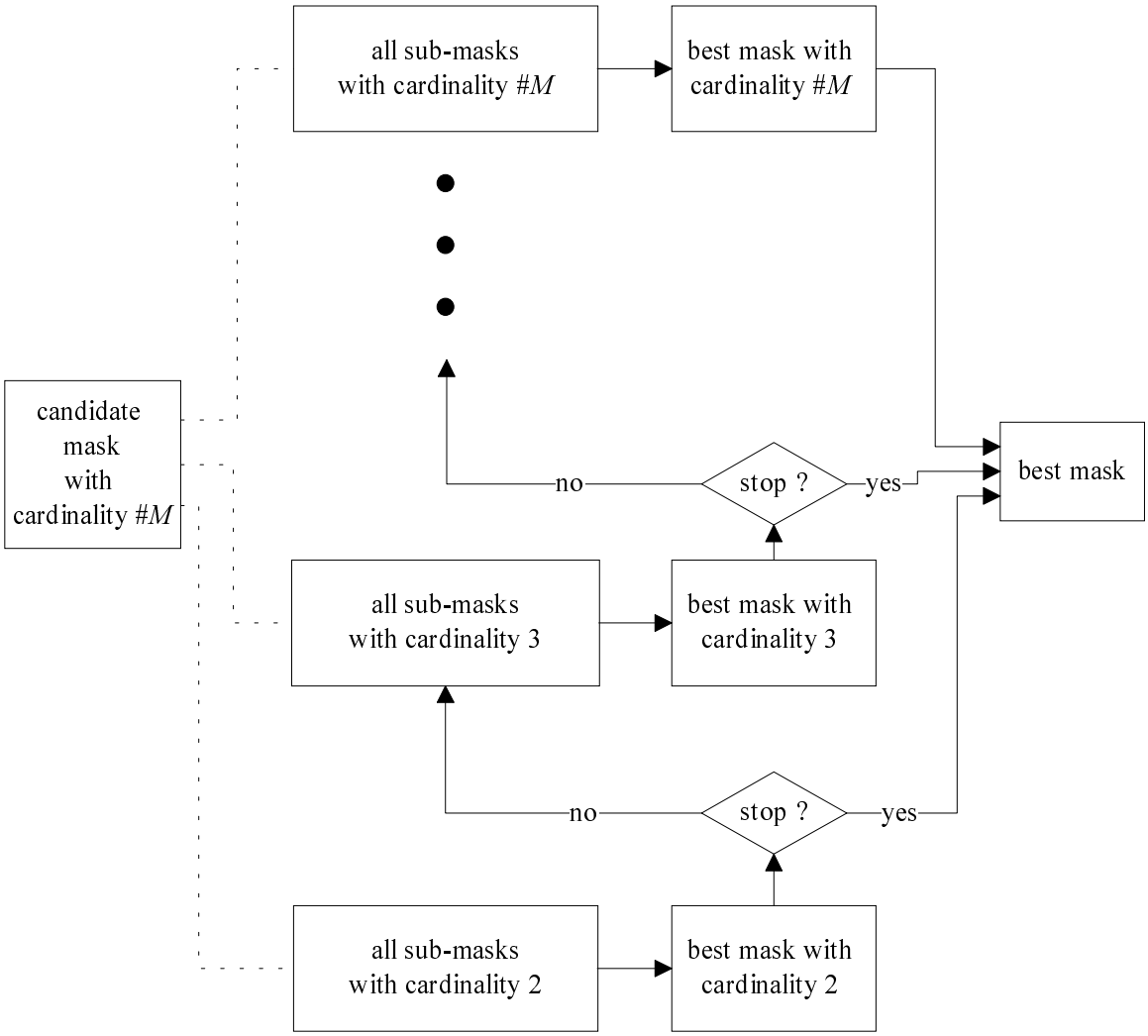


Figure 2.5 : Optimal mask search in SAPS-II

The drawback, however, is computing time. The time-complexity of the search process is nearly exponential because all combinations of masks with two input entries, three input entries, ... from the candidate mask are evaluated (see also chapter 4).

A general trend in the search is, that masks with a low number of entries generally have a low degree of determinism and a low degree of complexity. Of course, it is always possible that an underlying highly deterministic relation holds for a certain system, but in general this is quite rare. Hence, the next thoughts are about the general case. Thus, for (one-output) masks with a low number of *m*-inputs, or cardinality, one has a low quality due to the large degree of uncertainty, while for masks with a large number of *m*-inputs (high cardinality) one has a low quality due the higher complexity, e.g., observation ratio. This proves again that complexity and degree of determinism are conflicting measures.

Summarising, it can be stated that less complex masks (with regard to the number of *m*-inputs) have a low quality that will gradually increase when the masks become more complex,

but will start decreasing again. The tricky problem is that the optimum cannot be guessed beforehand. Chapter 4 elaborates more on these issues.

2.3.4 Forecasting

Shifting a mask over the data allows forecasting. The mask selects relevant data values from the data list by selecting elements for which the entries in the mask are not zero (see chapter 3 for an example). With this given mask, usually the optimal mask, one can forecast easily by using the state-transition matrix as a look-up table for (new) data (in the state space) for which the inputs are given. Another method is to use a nearest neighbour method for forecasting.

Forecasting with the state-transition matrix

The state-transition matrix acts as predictive model: it relates the inputs on certain (past) time instances to the output value or outcome on the reference time instance, denoted by O_t. One may speak of a one-step prediction. With each one-step prediction a probability is attached. When one wants to predict further into the future, the state-transition matrix is applied recursively. This can be expressed formally by considering a given outcome sequence $O = O_{t+1}O_{t+2}\cdots O_{t+n}$. The probability for having this sequence is given by

$$P(O) = \prod_{i=1}^{n} P(O_{t+i}) \qquad (2.14)$$

At a certain moment the probability of occurrence of an outcome sequence may drop below a (pre-defined) threshold. From this moment on, the prediction is too uncertain to be used. Hence, SAPS limits its own prediction horizon.

If the forecasting procedure encounters an *m*-input state (overall generating state) that has not been observed before, the forecasting process comes to a halt. A second best (or third, fourth, etc) best state-transition matrix can then be investigated to see if it can do a forecast (for an example, see appendix B). In SAPS-II, the second, third, etc. comes from a level with another cardinality.

In case of fuzzy recoding an extra process, called regeneration is needed to restore the quantitative values of the predicted outcomes. Regeneration is similar to defuzzification. In that process the membership values and the side function are used to specify more precisely the exact value of the output to be predicted. The confidence measure can be used in a similar way to limit the prediction horizon.

Forecasting with a nearest neighbours method

In later versions of SAPS-II, a five nearest neighbours (5NN) method is used for forecasting, [Mugica and Cellier 1993 ; Cellier 1996]. Suppose there are *n m*-inputs ($v_{i,j}$, $j = 1, ..., n$) and one output (out_i), then a row *i* in the I/O matrix will look like (compare with Table 2.6)

$$v_{i,1}v_{i,2}\cdots v_{i,n} \rightarrow out_i \qquad (2.15)$$

or in vector notation

$$\vec{v}_i \rightarrow out_i \qquad (2.16)$$

Each recoded variable (be it an *m*-input or output) consists of a set value, a membership value, and a side value. When doing a prediction one looks for similar records in the I/O matrix among the *m*-inputs. Hence, of interest at this moment are the set, membership and side val-

ues of the *m*-inputs. Let us denote for each *m*-input $v_{i,j}$, its set value by $class_{ij}$, its membership value by $Memb_{i,j}$, and its side value by $side_{i,j}$. With these values, a normalised $p_{i,j}$ value is computed via

$$p_{i,j} = class_{i,j} + side_{i,j} * (1 - Memb_{i,j})$$

For each row *i* represented by equation (2.16), denote the corresponding *m*-input vector, which is our point *i* in the measurement space, by $\vec{p}_i = (p_{i,1}, p_{i,2}, \cdots, p_{i,n})$ if there are *n* *m*-inputs. Hence, a new list of records where each record *i* is of the form

$$\vec{p}_i \rightarrow (class_i, Memb_i, side_i),$$

is created where the right hand side object represents the recoded output out_i.

When a new input vector $\vec{v}$ is detected for which the outcome is not known, i.e.,

$$\vec{v} \rightarrow out? \tag{2.17}$$

and the corresponding *out* has to be predicted, its *m*-input vector $\vec{p}$ is determined and compared with the list of all known $\vec{p}_i$'s. This is done via the Euclidean distance (L_2 norm), which is computed via

$$d(\vec{p}_i, \vec{p}) = \sqrt{\sum_{j=1}^{n} \left(p_{i,j} - p_j \right)^2} \tag{2.18}$$

The nearest point is used to forecast the class and side value of the output, i.e.,

$$class, side \leftarrow \underset{i}{\operatorname{argmin}}\, d(\vec{p}_i, \vec{p}) \tag{2.19}$$

The membership value is determined by a weighted sum of the five nearest neighbour membership values. It is given by

$$Memb(out) = \sum_{i \in \{5N\}} w_{rel,i} Memb(out_i) \tag{2.20}$$

where {5N} is the set containing the five nearest neighbours (or less than five if no five could be found). The normalised relative weights $w_{rel,i}$ are based on some inverse proportionality with the distances[11] via absolute weights. E.g., they can be computed by

$$w_{abs,i} = \frac{d_{\max} - d_i}{d_{\max}}$$

for $d_i \in \{5N\}$. Other manners can be devised for computing the absolute weights, [Mugica and Cellier 1993 ; Nebot 1994 ; de Albornoz 1996 ; Herrera 1999]. For example, one could take $w_{abs,i} = 1/d_i$. If a point coincides with the new point, i.e., $d_i = 0$ then this point is taken as the new point. The sum of the absolute weights over {5N} can then be used as a normalisation factor to obtain the relative weight, which are all between 0 and 1, i.e.,

$$w_{rel,i} = \frac{w_{abs,i}}{s_w},$$

where

[11] How they are computed can be found in [Cellier 1996].

$$s_w = \sum_{i \in \{5N\}} w_{abs,i}$$

Equations (2.19) and (2.20) determine the new point completely. Via an inverse process of recoding, called regeneration (a kind of defuzzification), one can reconstruct a (continuous) data point for equation (2.17) (For example, solve equation (2.3) for *x*) Using this method on the known data always gives a perfect forecast, because one distance will always be zero. Prediction errors can be obtained via similarity and proximity measures, [Herrera 1999].

2.4 Conclusion

A tool called SAPS, which is based on GSPS, has been designed to facilitate non-parametric system identification. In SAPS, restriction to a causal (one-output) mask is made in the context of a generative system. Inputs and generating states are called *m*-inputs, while generated states are called *m*-outputs. Time is the only support variable. Much attention has been given to the recoding process (source and data system) and to the evaluation and construction of masks (behavioural system). A sensible evaluation requires a trade-off between a normalised Shannon entropy and a complexity measure, which is cast in an evaluation function. The search algorithm that finds the best mask (i.e., the one with the highest quality) is of an exhaustive nature. In SAPS, forecasting can be done via a state-transition matrix or via a nearest neighbour method. Both allow for a limitation of the forecasting horizon via prediction error determination.

The fuzzy recoding used in SAPS cannot be placed easily at the source or data system level. If fixed-sized intervals are used, then is belongs to the source system (it is a combination of observation and abstraction function. If uniform-recoded intervals are used, then it belongs to the data system (is it an observation channel that relies on data?). Fortunately, this has no implications for the remainder of this thesis.

References

Åström K.J., Wittenmark B. [1997], Computer-Controlled Systems. Theory and Design, Prentice-Hall, New Jersey, 1997.

Cellier F.E. [1987], "*Qualitative Simulation of Technical Systems by Means of the General System Problem Solving Framework*", International Journal of General Systems, 13(4), p. 333-344, 1987.

Cellier F.E. [1991], Continuous System Modelling. Springer-Verlag, New York, 1991.

Cellier F.E., and Yandell D.W. [1987], "*SAPS-II: A New Implementation of the Systems Approach Problem Solver*", International Journal of General Systems, 13(4), p. 307-322, 1987.

Cellier F.E., Nebot A., Mugica F., and de Albornoz A. [1996], "*Combined Qualitative/Quantitative Simulation Models of Continuous-Time Processes Using Fuzzy Inductive Reasoning Techniques*", International Journal of General Systems, 24, 1-2, p. 95-116, 1996.

de Albornoz A. [1996], Inductive Reasoning and Reconstruction Analysis: Two Complementary Tools for Qualitative Fault Monitoring of Large-Scale Systems. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1996.

Herrera J.L. [1999], Time Series Prediction Using Inductive Reasoning Techniques. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1999.

Law A., Kelton D. [1990], "*Simulation Modelling and Analysis*", 2nd edition, Mc Graw-Hill, New York, 1990.

Li D., Cellier F.E. [1990], "*Fuzzy Measures in Inductive Reasoning*", Proceedings of the 1990 Winter Simulation Conference, New Orleans, LA, p. 527-538, 1990.

Mugica F. [1995], Diseño Sistemático de Controladores Difusos Usando Razonamiento Inductivo. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1995 (in Spanish).

Mugica F., and Cellier F.E. [1993], "*A New Fuzzy Inferencing Method for Inductive Reasoning*", Proceedings International Symposium on Artificial Intelligence, Monterrey, Mexico, p. 372-379, 1993.

Nebot A. [1994], Qualitative Modelling and Simulation of Biomedical Systems Using Fuzzy Inductive Reasoning. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994.

Uyttenhove H.J. [1978], Computer-Aided Systems Modelling: An Assemblage of Methodological Tools for Systems Problem Solving. Ph.D. thesis, School of Advanced Technology, State University of New York at Binghamton, 1978.

Van Welden D.F., Vansteenkiste G.C. [1994], "*SAPS-ST: A Testbed For Incremental Research on GSPS*", Proceedings of the 1994 European Simulation Multiconference, Barcelona, Spain, June 1-3, p. 507-513, 1994.

Chapter 3

Formalising Model Construction in SAPS with Hidden Markov Models

3.1 Introduction

The link between SAPS and finite memory machines is outlined in this chapter. With this in mind, states and state-models for black-box systems are defined. An extension to a stochastic system can then be formalised via a hidden Markov model. The corresponding formalisation, which is situated at the behavioural level in GSPS, proves that the state-transition construction procedure followed in SAPS can be seen as a newly defined hidden Markov model problem type. Two important side-results emerge. The first shows that correlation in the system memory does not play a role for the system identification paradigm on which SAPS resides. The second shows that the outputs are conditionally independent. The latter is a necessary condition for the methodology in part 2 of this thesis.

This chapter rigorously justifies and clarifies the underlying paradigm in SAPS. The correspondence with finite memory machines also allows a definition of what is meant by complex, black-box systems. A simplification to SISO (Single Input Single Output) systems is done to simplify explanations when there is no conceptual difference with the MIMO (Multiple Input Multiple Output) case.

3.2 State models and input-output models in SAPS

SAPS is best at identifying a time-invariant (stationary) causal dynamical system. Its applicability as state model for time-invariant (causal) systems will be discussed in this chapter. The concept of state models is well known in the theory of dynamical systems. State variables allow specifying in a unique way the state of the system at each time instance *t*. For a directed system the output is completely specified if one knows the input and the state on the same time instance. For causal discrete[1] time-invariant (SISO) systems the mathematical description is given by

$$\begin{aligned} x_{k+1} &= f(u_k, x_k) \\ y_k &= g(u_k, x_k) \end{aligned} \tag{3.1}$$

with

- u_k the input at time instance *k*,
- x_k the state at time instance *k*,

[1] Assume periodic sampling to have happened.

- y_k the output at time instance k,
- f the state-transition function,
- g the output or response function.

Note: Sometimes, $u(k)$ is used instead of u_k. It is especially advisable when having different inputs, e.g., $u_1(k)$ and $u_2(k)$, or when applying operators on k, e.g., a lag operator.

The first equation is called the state-transition equation and the second the output equation. For specified finite sets of stimuli and states the state-transition equation partially[2] describes a Finite State Automaton (FSA) [Carroll and Long 1989]. If one encompasses the output function too then the discrete system described by equations can be compared with a Mealy machine or Finite State Transducer (FST). A state transition diagram for a Mealy machine is illustrated in Figure 3.1. It shows that if an input $u(k)$ is applied to a state $x(k)$ this results in the state going to a next state $x(k+1)$. During this transition, state $x(k)$ sends out an output $y(k)$.

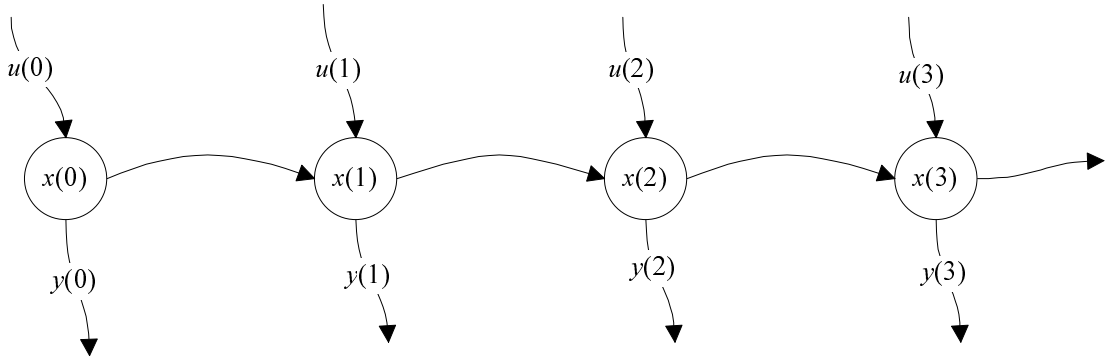


Figure 3.1 : State transitions for a Mealy machine

The abstract theory of automata is oriented to problems in which the UC-structure is not considered. Such problems are, for instance, the determination of the functions f and g, or the ST-structure for a given behaviour. Hence, seeking a relationship with finite state automata (or extensions of them) makes sense.

In Input/Output models, however, only observed input/output behaviour stemming from measured data is available (black box systems). Hence, a state has to be defined based on this available information. The state summarises past information into one instance in time, e.g., in the output function the term x_k summarises input information $u_{k-1}, u_{k-2}, \cdots, u_{k-\infty}$ for an input-output model. In addition, a state can also be constructed with past outputs $y_{k-1}, y_{k-2}, \cdots$. As an illustration, consider the following (linear) infinite-lag regression model of the form [SAS/ETS 1993] (a_i and w are coefficients):

$$y(k) = a_0 + a_1\left(u(k) + wu(k-1) + w^2u(k-2) + \cdots\right) + \varepsilon(k)$$

It reduces after some manipulation and with ($b_0 = a_0(1-w), v(k) = \varepsilon(k) - w\varepsilon(k-1)$) to

$$y(k) = b_0 + wy(k-1) + a_1u(k) + v(k)$$

This illustrates the reduced number of parameters by introducing an auto-regressive term.

Thus, taking into account a previous output reduces the needed number of lagged inputs. This principle might also be applied in general, obtaining

[2] An initial state is implicit in the state-transition equation, but no set of final states is defined.

$$x_k = y_{k-1}, y_{k-2}, ..., y_{k-m}, u_{k-1}, ..., u_{k-n}$$

The output equation of (3.1) can then generally be written as:

$$y_k = h(u_k, y_{k-1}, y_{k-2}, ..., y_{k-m}, u_{k-1}, ..., u_{k-n}) \qquad (3.2)$$

However, it still remains to be investigated what is the right mix of number of past outputs and number of (past) inputs, i.e., the determination of *m* and *n*.

Equation (3.2) is used in directed generative systems (see equation (1.16)) in GSPS where one initially considers the input at the reference time instance and a finite number of past outputs and inputs (generating states). A corresponding mask for equation (3.2) can be readily written down (see chapter 2). For reasons of (computational) complexity, *m* and *n* should not be too large. The main purpose in GSPS is to find a meaningful and concise (low mask cardinality) mix of the arguments in equation (3.2) according to certain criteria.

Comparing equation (3.2) with a finite-memory automaton reveals that GSPS tries to identify the latter. The finite-memory approach is suitable if the behaviour is given in the form of an activity. However, GSPS allows some degree of non-determinism, so it would be more correct to compare it with a probabilistic version of a finite-memory automaton. Taking into account the quantisation of the variables and that the next state only depends on the current state and the inputs, one thus arrives at 'controlled Markov chains', a term mentioned in [Bagchi 1993]. However, to the author's knowledge no hidden controlled Markov chain exists with accompanying techniques for system identification. Thus, another solution has to be sought.

Fortunately, chapter 2 showed that SAPS-II does not distinguish between inputs, past inputs and outputs, i.e., the case of a generative system (with only generating and generated states) applies. In fact, this looks very similar to the transformation of a Mealy machine to a Moore machine via the theorem of Gill and Bloh, [Gécseg and Peák 1972]. Therefore, a new line of thought based on the perception that it is possible to consider a new general state given by

$$x_k^* = u_{k,} x_k$$

will be followed. Hence, following a similar reasoning, equation (3.2) can be rewritten as

$$y_k = g'(x_k^*)$$

where x_k^* is a generating state that replaces the argument $u_k, y_{k-1}, y_{k-2}, ..., y_{k-m}, u_{k-1}, ..., u_{k-n}$ and y_k as the generated state. Therefore, in SAPS-II the new set of equations becomes

$$\begin{aligned} y_k &= g'(x_k^*) \\ x_{k+1}^* &= f'(x_k^*) \end{aligned} \qquad (3.3)$$

The final step consists in relaxing the deterministic requirements on the state-transition and the output function (the state-transition function will prove to be irrelevant in the SAPS system identification paradigm). Accordingly, in going from a deterministic to a stochastic version there is no unique consequent to every state anymore and some probability distribution is needed. The sought probabilistic alternative should apply for SAPS-II, see equation (3.3). Such a variant can be found in the theory of hidden Markov models.

3.3 Hidden Markov models

A countable sequence of random variables $\{X_n; n \geq 0\}$ is called a general discrete Markov chain if each $X_n \in Z$ (Z is the set of integers, i.e.…,-2,-1,0,1,2,…), and iff the Markov condition is satisfied, i.e., for any positive integer *n*, any sequence $\{i_k \mid 0 \leq k \leq n\}$ and $j \in Z$ one has

$$P\{X_{n+1} = j \mid X_0 = i_o, X_1 = i_1, \cdots, X_n = i_n\} = P\{X_{n+1} = j \mid X_n = i_n\} \qquad (3.4)$$

The random variable X_n is called the state of the chain at time *n* while X_0 is called the initial state. If the probabilities $P\{X_{n+1} = j \mid X_n = i_n\}$ do not depend on *n*, then the general discrete time Markov chain is called stationary or simply a discrete (time) Markov chain [Tijms 1994; Hock 1996].

The Markov condition indicates that the future depends on the past only through the present. The probabilities $P\{X_{n+1} = j \mid X_n = i_n\}$ can be considered as elements of a matrix *A* such that

$$A_{i,j} = \{P(X_{n+1} = j \mid X_n = i)\} \qquad (3.5)$$

Matrix *A* is a stochastic matrix. It satisfies the two requirements

1. $\forall i, j \in \mathrm{Z}: A_{i,j} \geq 0$
2. $\forall i \in Z: \sum_{j \in Z} A_{i,j} = 1$

Markov models in which each state corresponds to one observable event are too restrictive to be applicable to many problems of interest. Hence, an extension is made where the observation is a probabilistic function of the state. The resulting hidden Markov model is a doubly embedded stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observations (see Figure 3.2).

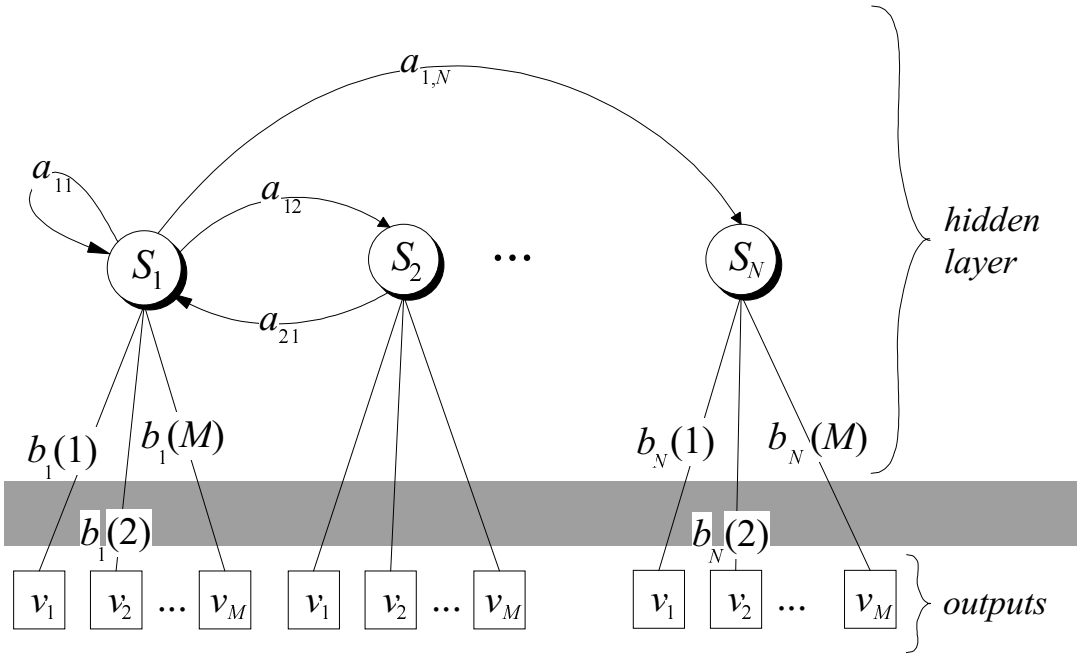


Figure 3.2 : A general hidden Markov model

An observation sequence is denoted by

$$O = O_1 O_2 \cdots O_T ,$$

where T is the number of observations.

Each observation O_t, (t = 1,…, T) stems from an alphabet, denoted by V (see below).

The hidden Markov model is characterised by

- A number of states in the model, say N states, where often some physical significance is attached to the states. The individual states are given by $S_i, i = 1, \cdots, N$ and the state at time t is denoted by q_t.
- The initial state distribution $\Pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i), \qquad 1 \le i \le N$
- The state transition probability distribution/matrix, denoted by $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$ (Markov condition)
- A number of distinct observation symbols per state. Usually, the same discrete alphabet is used. The alphabet size is denoted by M. The individual symbols are given by $V = \{v_1, v_2, \cdots, v_M\}$
- The observation symbol probability distribution/matrix in state S_i ; $B = \{b_i(k)\}$, where $b_i(k) = P(v_k \text{ at } t \mid q_t = S_i), \qquad 1 \le i \le N, \qquad 1 \le k \le M$

Hence, a complete specification of a hidden Markov model involves specifying the model parameters N and M, the symbol set V, and the specification of the three probability distributions A, B and Π. In compact form the complete parameter set is denoted by $\lambda = (A, B, \Pi)$.

The roles of these parameters are nicely illustrated in the algorithm for generating an observation sequence:

- initialisation :
  1) set t = 1
  2) choose initial state q_1 according to Π
- loop :
  3) for t = 1 to T do:
  {
  3a) in state q_t put $O_t = v_k$ according to B
  3b) transit to new state q_{t+1} according to A
  }

<u>An example of a hidden Markov model: urn and ball model</u>

The urn and ball model illustrates a hidden Markov model for which one has chosen N states; each state corresponds to an urn. In each urn there are, say, M distinct coloured balls (there may be more than M balls). An observation is now generated as follows:

According to some random process (Π), an initial urn was selected and from that urn a ball is selected at random, (B). The colour is recorded, O_t, and the ball put back in the same urn. A

new urn is then selected according to some process associated with the current urn, (*A*) (otherwise, this is not Markov) and the ball selection procedure is repeated until, say, *T* colours are observed.

The random process of selecting a new urn (could be the same urn) based on the current urn is the Markov process (probabilities denoted by a_{ij}; *i*: current urn, *j*: new urn). The process of selecting a ball from the selected urn is multinomial, where

- the probabilities are denoted by $b_i(color)$,
- *i* stands for the index of the urn,
- usually colour is replaced by an integer resulting from a labelling mapping (bijection between colours and *M* integers).

To fix ideas, consider three urns (*N*=3) and three colours (not necessarily 3 balls) in each urn (*M* = 3). The colours are red (R), green (G) and blue (B). The Markov chain is considered left-right and is given in Figure 3.3.

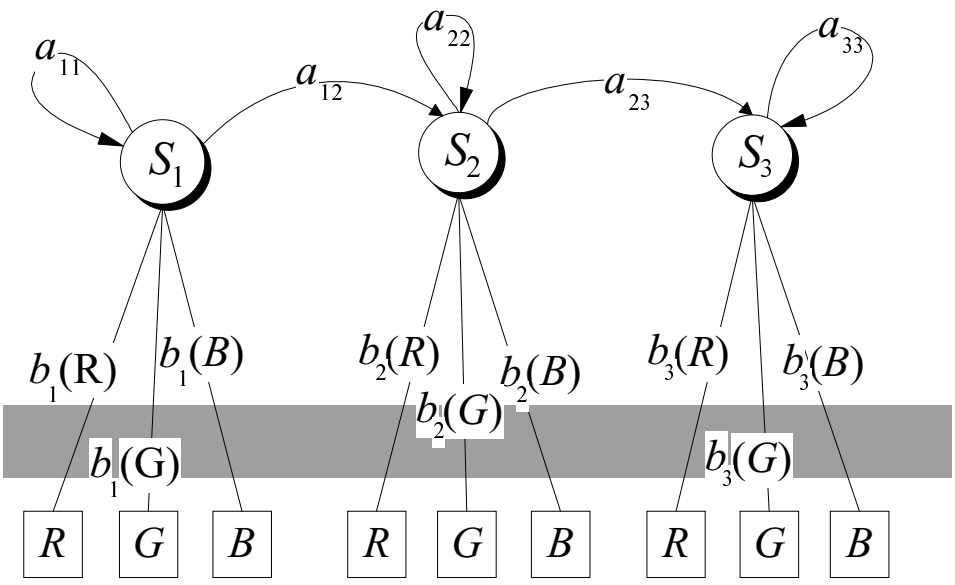


Figure 3.3 : An example of a hidden Markov model for the urn example

The initial state distribution is given by

$$\Pi = (0.7 \quad 0.2 \quad 0.1)^T$$

The state-transition matrix is known and given by

$$A = \begin{pmatrix} 0.2 & 0.8 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$

The observation-distribution (conditionally on the states) by

$$b_1(R) = 0.4 \quad b_2(R) = 0.1 \quad b_3(R) = 0.3$$
$$b_1(G) = 0.4 \quad b_2(G) = 0.2 \quad b_3(G) = 0.2$$
$$b_1(B) = 0.2 \quad b_2(B) = 0.7 \quad b_3(B) = 0.5$$

An observation sequence could be for example *O* = (*R*,*G*,*G*,*B*,*R*,*B*)

3.4 Construction of a predictive model in SAPS

The next example starts from a binary recoded activity matrix given in Table 3.1. *t* indicates discrete time instances, the index *i* will be used for state labelling.

t	u_1	u_2	u_3	u_4	*y*
1	0	1	0	1	0
2	1	1	1	1	1
3	0	1	0	1	1
4	0	1	1	0	0
5	1	0	0	0	0
6	1	1	1	1	1
7	1	0	1	0	1
8	0	1	0	1	0
9	0	0	1	1	1
10	1	1	1	1	0
11	0	1	1	1	0
12	1	1	0	1	0
13	1	1	1	1	0
14	0	1	1	0	1
15	0	0	1	0	1
16	1	1	1	0	0
17	1	1	1	1	0
18	0	0	1	0	0
19	0	1	1	1	0
20	1	0	1	0	1

Table 3.1 : A list of recoded data records

An example of a mask is presented in Table 3.2.

	u_1	u_2	u_3	u_4	*y*
t-2	-1	0	0	0	0
t-1	0	-1	0	0	0
t	0	0	-1	0	1

Table 3.2 : A mask

The entry '1' gives the location of the output or generated state, i.e., $y(i)$. Entries that are '-1' stand for relevant generating states (inputs, past inputs, and past outputs). Zero entries indicate irrelevant entries for the dependency relation between output and generating states.

The mask with memory depth 2 in Table 3.2 stands for

$$y(t) = \tilde{f}(u_1(t-2), u_2(t-1), u_3(t))$$

with $\tilde{f}$ a qualitative function (which will later be associated with a probability-observation matrix).

State-observation vectors[3] for a mask are generated by sliding the mask over the data (see Table 3.1 where the shaded area shows the position of the mask at t = 9). $\vec{x} = (x_1, x_2, x_3)$ forms the state and y the output. The result is depicted in Table 3.3.

	state S_i			output v_k	state labelling
t	$x_1 = u_1(t\text{-}2)$	$x_2 = u_2(t\text{-}1)$	$x_3 = u_3(t)$	y	state number i
1			0	0	
2		1	1	1	
3	0	1	0	1	3
4	1	1	1	0	8
5	0	1	0	0	3
6	0	0	1	1	2
7	1	1	1	1	8
8	1	0	0	0	5
9	1	1	1	1	8
10	0	0	1	0	2
11	0	1	1	0	4
12	1	1	0	0	7
13	0	1	1	0	4
14	1	1	1	1	8
15	1	1	1	1	8
16	0	0	1	0	2
17	0	1	1	0	4
18	1	1	1	0	8
19	1	0	1	0	6
20	0	1	1	1	4

Table 3.3 : State-observation matrix

[3] Newly introduced terminology, consistent with the hidden Markov chain approach, is denoted in italics.

This matrix is built up by state-observation records denoted formally by $\{q_t = S_i, v_k\}$. In the example, the recoding is binary, thus $V=\{v_1,v_2\}$ with $v_1 = 0$, $v_2 = 1$. The headers of the table are extended with the appropriate notations.

In literature concerning SAPS, Table 3.3 is called an I/O model, but the term *state-observation matrix*[4] is more appropriate for it really describes what constitutes a row of the matrix. The last column in the table is usually not written, but it is needed in order to draw the state transition diagram in Figure 3.4.

The next step is to compress the state-observation matrix. The compression is reached by summarising the state-observation records that are the same and by computing their relative frequency and thus their probability of occurrence (see Table 3.4). In SAPS-II, this is called the basic behaviour.

S_i			v_k	$\#(v_k, q_t = S_i)$	$P(v_k, q_t = S_i)$	
x_1	x_2	x_3	y	state-observation frequency	State-observation probability	position of mask
0	0	1	1	1	0.055556	6
0	0	1	0	2	0.111111	10, 16
0	1	0	1	1	0.055556	3
0	1	0	0	1	0.055556	5
0	1	1	0	3	0.166667	11, 13, 17
0	1	1	1	1	0.055556	20
1	0	0	0	1	0.055556	8
1	0	1	0	1	0.055556	19
1	1	0	0	1	0.055556	12
1	1	1	0	2	0.111111	4, 18
1	1	1	1	4	0.222222	7,9,14,15

Table 3.4 : Compressed state-observation matrix

The *compressed state-observation matrix* contains the *state-observation probabilities* $P(v_k, q_t = S_i)$.

A stochastic matrix is obtained by rewriting the above in the form of a 'state transition' matrix. In Table 3.5 the 'state transition matrix' is given with the frequency of the generating part of the state record and its associated generating state probability. As will be shown soon, this matrix is in fact an *observation-distribution matrix.*

[4] In the state-observation matrix each record has a frequency of 1

	state frequency $\#(q_t = S_i)$	x_1	x_2	x_3	y 0	y 1	generating state probability $P(q_t = S_i)$
	0	0	0	0	?	?	0.000000
	3	0	0	1	0.67	0.33	0.166667
	2	0	1	0	0.50	0.50	0.111111
	4	0	1	1	0.75	0.25	0.222222
	1	1	0	0	1.00	0.00	0.055556
	1	1	0	1	1.00	0.00	0.055556
	1	1	1	0	1.00	0.00	0.055556
	6	1	1	1	0.33	0.67	0.333333
total frequency	18						

Table 3.5 : 'State transition' matrix

The probabilities in the bordered zone are conditional probabilities; i.e., given a generating state what is the probability of a certain value of the output. The conditional output is given by the ratio of the corresponding state-observation frequency and the state frequency. Looking more closely at the bordered stochastic matrix in Table 3.5, shows that it gives the probability of observing a certain output symbol given a generating state or *m*-input, that is

$$P(y = k \mid \vec{x} = S_i) = P(v_k \text{ at } t \mid q_t = S_i) = b_i(k) \qquad (3.6)$$

which is semantically consistent with the hidden Markov model viewpoint. Using this to make Table 3.5 more explicit one obtains Table 3.6.

i	$\#(q_t = S_i)$	S_i			$v_1 = 0$	$v_2 = 1$	$P(q_t = S_i)$
1	0	0	0	0	$b_1(v_1) = ?$	$b_1(v_2) = ?$	0
2	3	0	0	1	$b_2(v_1) = 0.67$	$b_2(v_2) = 0.33$	0.166667
…	…						
8	6	1	1	1	$b_8(v_1) = 0.33$	$b_8(v_2) = 0.67$	0.333333

Table 3.6 : Observation-distribution matrix (not completely shown)

The output symbols are simply the indices, i.e., $v_i = i$, or a shift of them, e.g., $v_i = i - 1$, depending on convention. The estimation of the conditional probabilities is based on frequency counts (the next equation[5] can also be found in problem type 3 of hidden Markov chains [Rabiner 1989]:

$$\bar{b}_i(k) = \frac{\text{expected \# of times in } S_i \text{ and observing } v_k}{\text{expected \# of times in } S_i} = \frac{\#(v_k, q_t = S_i)}{\#(q_t = S_i)} \quad t = 1, \cdots, T$$

[5] bar denotes estimate, # denotes 'number'

This is consistent with

$$b_i(k) = P(v_k \text{ at } t \mid q_t = S_i) = \frac{P(v_k, q_t = S_i)}{P(q_t = S_i)} \qquad (3.7)$$

where in SAPS direct estimates for the numerator and denominator are used.

3.5 Applying Hidden Markov models to SAPS

Table 3.3 shows the sequence of states after applying the mask in Table 3.2 to the recoded data presented in Table 3.1. This can be visualised in a state transition diagram. Taking further into account the state dependent output probabilities, a hidden Markov model can be drawn as in Figure 3.4.

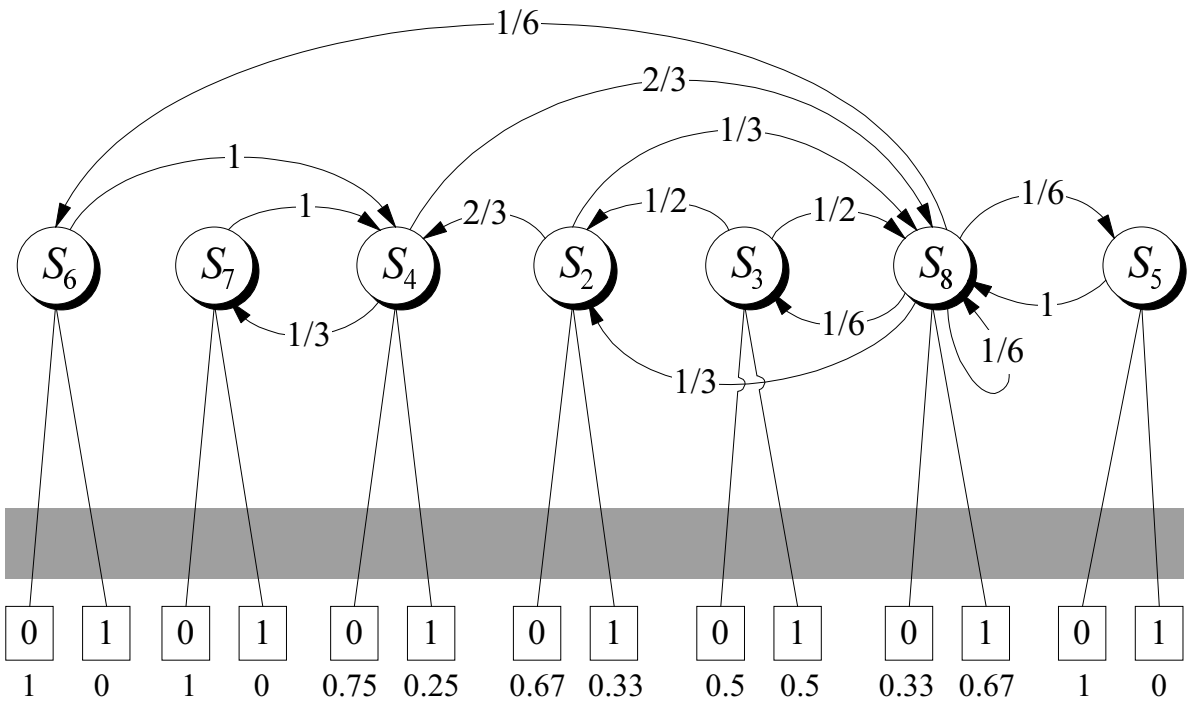


Figure 3.4 : Hidden Markov model for the example in Table 3.1 scanned by the mask in Table 3.2

This shows us that two approaches are possible:

1) modelling the output probabilities conditional on the states and accepting the state sequences as given

2) modelling the state transitions as well

The first approach is used in SAPS where one has no interest in the (auto or cross)correlation structure of the *m*-inputs (this can be seen as source system level assumptions about generating variables and inputs in GSPS). The *m*-inputs are given a priori and for prediction one can rely on their availability for each prediction. Hence, the correlation structure is not necessary for prediction. Consider for example a hypothetical system where one looks at the growth of plants (output) in relation with the inputs rain, sunshine, and fertilisation (see Figure 3.5).

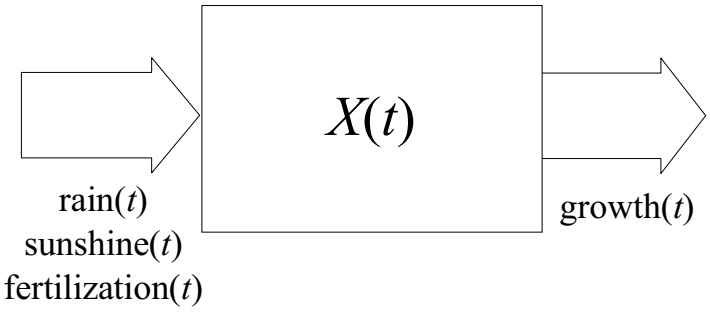


Figure 3.5 Example of plant growth

It is to be expected that when it rains, there will be no sunshine and vice versa. Thus, rain and sunshine must be correlated. However, for prediction this does not matter because rain and sunshine on future time instants will be given (with the right correlation automatically). Thus, correlation implicitly present in the inputs will not affect prediction of the output. With regard to Figure 3.4, one could say that each state S_i is given when necessary for prediction. Hence, one can treat the states as independent with regard to the output prediction, because the outputs are, when conditioned on the states, independent (see sections 3.6.1 and 3.7).

The situation becomes different if one also would like to do predictions *without* being able to rely on newly given *m*-inputs. Then, one has to predict the next state to occur and based on this state, one may do a prediction for the output. Thus, correlation structures between *m*-inputs are taken into account via a state-transition matrix to predict the most likely next state S_{i+1}. The corresponding probability for the chosen state-transition has to be taken into account when computing probabilities for the predicted output. In this approach, one is in fact modelling neutral systems via discrete stochastic processes[6].

The second approach will not be elaborated much further in this thesis, although it can be put in the framework of hidden Markov chains if a next state only depends on the current state as assumed in Figure 3.4. However, it is demonstrated in much more detail in section 3.7 that the first approach, which is used in SAPS and which models the output probabilities conditional on the states for a given state sequence, is a new hidden Markov problem type.

3.6 Basic problems to solve in hidden Markov Chains

The problems encountered in hidden Markov models are now described according to their increasing degree of difficulty. The new type of problem to be defined later is most akin to type 1. Therefore, type 1 will be explained in some more detail. Type 2 and type 3 are well described in Rabiner's paper, [Rabiner 1989].

[6] An example of this approach can be found in Klir (1969) on page 120. There, a Markov chain is applied, although it is not mentioned as such.

3.6.1 Problem 1: Compute an Observation Sequence Given the Complete Model Description

Given an observation sequence $O = O_1 O_2 \cdots O_T$ and a model $\lambda = (A, B, \Pi)$, how to compute $P(O|\lambda)$?

Consider

$$P(O \mid \lambda) = \sum_{\text{all } Q} P(O, Q \mid \lambda)$$

and with

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda).P(Q \mid \lambda)$$

one has

$$P(O \mid \lambda) = \sum_{\text{all } Q} P(O \mid Q, \lambda).P(Q \mid \lambda).$$

Hence, enumerate every possible state sequence of length T and consider one of them, e.g. $Q = q_1 q_2 \cdots, q_T$. Then, under statistical conditional independence on the states the first factor is given by

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda).$$

This equation is a more rigorous formalisation of equation (2.14), which shows the prediction horizon. To have a more explicit representation, write

$$\begin{aligned} P(O \mid Q, \lambda) &= P(O_1 O_2 \cdots O_T \mid q_1 q_2 \cdots q_T, \lambda) \\ &\overset{\text{cond. stat. indep.}}{=} P(O_1 \mid q_1, \lambda) P(O_2 \mid q_2, \lambda) \cdots P(O_T \mid q_T, \lambda) \\ &= b_{q_1}(O_1) b_{q_2}(O_2) \cdots b_{q_T}(O_T) \end{aligned}$$

for this specific state sequence. The second factor is given by

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

and after substitution and rearrangement of the factors, one finds

$$P(O \mid \lambda) = \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T).$$

The problem with this solution is the number of computations involved. There are approximately N^T state sequences and for each state sequence roughly $2T$ calculations, making a total of $2TN^T$; e.g. 3 states, observation length 50, then approximately $100 * 3^{50} \cong 10^{26}$ calculations. The simple approach is thus computationally not efficient. A much more efficient algorithm for computing $P(O|\lambda)$ can be devised if one uses forward variables [Rabiner 1989].

Remark that this problem type gives an indication of how well a given model matches a given observation sequence. The higher $P(O|\lambda)$ the better the match.

3.6.2 Problem 2: Uncover Hidden State Part and Search 'Correct' State Sequence

Given an observation sequence $O = O_1 O_2 \cdots O_T$ and a model $\lambda = (A, B, \Pi)$, how to choose $Q = q_1 q_2 \cdots, q_T$ which is optimal in some sense or explains best the observations?

The phrase 'in some sense' indicates that there are several optimisation criteria.

One approach is to choose the states that are individually most likely. This corresponds with maximising the expected number of correct individual states. The Viterbi algorithm can be used to find the single best state sequence for a given observation sequence. A problem that may arise is that if certain state transitions have zero probability some non-valid states may occur. Possible solutions could be to look for state sequences that maximise the number of correct pairs or triples, …

3.6.3 Problem 3: Optimise Model Parameters to Obtain a Best Description of Observation Sequence

How to adjust the model parameters to maximise $P(O|\lambda)$?

It is at this stage where one tries to induce the best models for real observed phenomena.

Unfortunately, there is no known analytical and *optimal* way of estimating the model parameters for any finite observation sequence as training data. There do exist, however, methods that choose model parameters such that they *locally* maximise $P(O|\lambda)$. A method for re-estimation of the parameters Π, A, and B can then be obtained. It can be proven (see [Rabiner 1989]) that the new model corresponds with a critical point, i.e. (local) maximum, or at least that it is better. In the latter case, one uses iteratively the thus obtained new model — while improving each corresponding $P(O|\lambda)$ — until some limiting point. This final model is the maximum likelihood estimate of the hidden Markov model.

3.7 A newly defined problem type for hidden Markov models

This problem type can be stated as "*Compute an observation matrix given the state sequence and the observation sequence*".

In the construction of a probability observation matrix (formerly state transition matrix), one starts from a given output sequence and a given state sequence (Table 3.3). The goal is to determine the probability-observation matrix that makes the observation sequence under the given generating state sequence most likely. Hence, the problem type can be formalised as follows:

> Given
>
> - an observation sequence $O_1 O_2 \cdots O_T$
> - and state sequence $q_1 q_2 \cdots q_T$,
>
> determine
>
> - $b_i(k) = P(v_k \text{ at } t \mid q_t = S_i), \qquad 1 \le i \le N, \qquad 1 \le k \le M$
> - such that $P(O_1 O_2 \cdots O_T | q_1 q_2 \cdots q_T)$ is maximal.

Note that this problem type has not been previously considered in hidden Markov models.

This problem type is applicable to the hidden Markov model example in this thesis. As the observations conditioned on the state sequence are independent, one can write

$$P(O_1 O_2 \cdots O_T \mid q_1 q_2 \cdots q_T) = P(O_1 \mid q_1 q_2 \cdots q_T).P(O_2 \mid q_1 q_2 \cdots q_T) \cdots P(O_T \mid q_1 q_2 \cdots q_T) \qquad (3.8)$$

Remark that for a given factor in the right hand side the output observed at time *t* only depends on the state the system is in at that time, i.e.,

$$P(O_t | q_1 q_2 \cdots q_T) = P(O_t | q_t) \qquad (3.9)$$

The maximisation can thus be rewritten[7] as the maximisation of

$$P(O_1|q_1).P(O_2|q_2) \cdots P(O_T|q_T) \qquad (3.10)$$

Example

Consider a 2-state model with a Bernoulli observation distribution; e.g., two coins where each coin represents a state. Then, one has for a given observation sequence *HHTTT* (*H*: head, *T*: tail) and given state sequence

$$q_1 q_2 q_3 q_4 q_5 = S_1 S_2 S_1 S_1 S_2$$

to determine

$$b_i(H) = P(H \text{ at } t \mid q_t = S_i) \text{ and } b_i(T) = P(T \text{ at } t \mid q_t = S_i) \text{ for } 1 \le i \le 2$$

such that

$$P(H|q_1).P(H|q_2).P(T|q_3).P(T|q_4) \ .P(T|q_5)$$

is maximal. Writing out the expression to be maximised:

$$P(H \text{ at } 1 \mid q_1 = S_1).P(H \text{ at } 2 \mid q_2 = S_2).P(T \text{ at } 3 \mid q_3 = S_1).P(T \text{ at } 4 \mid q_4 = S_1).P(T \text{ at } 5 \mid q_5 = S_2)$$

It can be rewritten as

$$b_1(H).b_2(H).b_1(T).b_1(T).b_2(T)$$

Using powers, this becomes

$$[b_1(H)]^1.[b_1(T)]^2.[b_2(H)]^1.[b_2(T)]^1$$

To maximise this expression, one needs to take into account the dependencies that are present among the different factors. Dependencies are conditional on the state. Hence, maximisation of the expression can be done for each state separately. Thus, maximise respectively

$$[b_1(H)]^1.[b_1(T)]^2 \quad \text{and} \quad [b_2(H)]^1.[b_2(T)]^1.$$

Taking into account the constraints

$$b_1(H) + b_1(T) = 1, \ b_2(H) + b_2(T) = 1$$

the respective maxima are

$$b_1(H) = \frac{1}{3}, b_1(T) = \frac{2}{3} \quad \text{and} \quad b_2(H) = \frac{1}{2}, b_2(T) = \frac{1}{2}.$$

[7] Notice the resemblance with the type 1 problem

Formalisation for the general case

For the general case, one has to maximise equation (3.10), i.e.,

$$P(O_1 \mid q_1 = S_i).P(O_2 \mid q_2 = S_j)\cdots P(O_T \mid q_t = S_k)$$

which allows for the possibility that a certain state appears more than one time, e.g. $q_i = q_j$. To simplify notation, rewrite this with observation probabilities, i.e.,

$$b_i(O_1).b_j(O_2)\cdots b_k(O_T)$$

Due to the state dependency, maximisation can be done per state, i.e., group per state for maximisation:

$$\max\left[\prod_{i=1}^{N} b_i(v_1)^{e_{i,1}}.b_i(v_2)^{e_{i,2}}\cdots b_i(v_M)^{e_{i,M}}\right] = \prod_{i=1}^{N}\max\left[b_i(v_1)^{e_{i,1}}.b_i(v_2)^{e_{i,2}}\cdots b_i(v_M)^{e_{i,M}}\right]$$

where the observation symbol notation is used and where $e_{i,1}, e_{i,2}, \cdots, e_{i,M}$ are parameters indicating how many times a certain symbol, $v_1, v_2, \cdots, v_M$, is observed for state S_i, i.e.

$$\begin{cases} e_{i,k} = \#(v_k, q_t = S_i) \\ \sum_{k=1}^{M} e_{i,k} = \#(q_t = S_i) \end{cases} \tag{3.11}$$

Thus, one has to maximise

$$\prod_{k=1}^{M} b_i(v_k)^{e_{i,k}}$$

under the constraint

$$\sum_{k=1}^{M} b_i(v_k) = 1.$$

Assuming for illustration purposes that there are three (non-zero) factors, then this boils down to maximising

$$b_i(v_1)^{e_{i,1}}.b_i(v_2)^{e_{i,2}}b_i(v_3)^{e_{i,3}}$$

subject to the constraint

$$b_i(v_1) + b_i(v_2) + b_i(v_3) - 1 = 0.$$

Applying Lagrange multipliers results in

$$b_i(v_1) = \frac{e_{i,1}}{e_{i,1}+e_{i,2}+e_{i,3}}, b_i(v_2) = \frac{e_{i,2}}{e_{i,1}+e_{i,2}+e_{i,3}}, b_i(v_3) = \frac{e_{i,3}}{e_{i,1}+e_{i,2}+e_{i,3}}$$

or rewritten, taking into consideration equation (3.11):

$$\begin{cases} b_i(v_1) = \dfrac{\#(v_1, q_t = S_i)}{\#(q_t = S_i)} \to P(v_1 \mid q_t = S_i) \\ b_i(v_2) = \dfrac{\#(v_2, q_t = S_i)}{\#(q_t = S_i)} \to P(v_2 \mid q_t = S_i) \\ b_i(v_3) = \dfrac{\#(v_3, q_t = S_i)}{\#(q_t = S_i)} \to P(v_3 \mid q_t = S_i) \end{cases}$$

It can be shown that this is a maximum via Taylor expansion and by looking at the second derivatives, which constitute a bilinear form. Hence, in general the maxima are given by

$$b_i(v_k) = \frac{\#(v_k, q_t = S_i)}{\#(q_t = S_i)} \to P(v_k \mid q_t = S_i) \qquad i = 1, \cdots, N \qquad k = 1, \cdots, M$$

These maxima are the same as originally computed with SAPS, see equation (3.7). Hence, SAPS solves newly introduced problem type for hidden Markov chains in the best (and simplest) way.

3.8 Conclusion

Candidate model construction in SAPS can be formalised as a newly defined problem type in the theory of hidden Markov models. The formalisation, which is applied at the behavioural system level, proves that the classical way to compute the conditional probabilities in a SAPS 'state-transition' matrix is optimal. Model order determination problems arising in SAPS are equally present in the theory of hidden Markov chains. As a spin-off result, the use of straightforward terminology shows that the term 'state transition matrix' appears to be a bit deceptive for it is in reality a probabilistic observation matrix. Other terms are proposed such as state-observation matrix and compressed state-observation matrix. They are more descriptive and consistent with the underlying structure of states and observations.

A new problem type strongly related to type 1 is found. Both types are fortunately analytically solvable. It is important to note that model order determination problems encountered in hidden Markov models are of the same kind as encountered in SAPS. They are:

- Models that are more complex are more capable of modelling (in the sense of giving a good internal fit) a series of observations. This corresponds to more and larger states in hidden Markov models, and to higher mask cardinality in SAPS.
- In type 3 problems, a local maximum is obtained. This is similar to the search for the 'right' mask in SAPS. An exhaustive search may become intractable for reasonably complex systems. This issue will be dealt with in detail in chapter 4.

Finally, as shown in the newly defined problem type, correlation among inputs are not of importance for the research done on directed systems in this thesis. Furthermore, the outputs are independent if conditioned on the generating states. This issue will be of vital importance when considering the whole new approach in part two of this thesis.

References

Bagchi, A. [1993], Optimal Control of Stochastic Systems. Prentice Hall, New York, 1993.

Carroll J. and Long D. [1989], Theory of Finite Automata (with an Introduction to Formal Languages). Prentice Hall, New Jersey, 1989.

Gécseg F. and Peák I. [1972], Algebraic Theory of Automata, Akadémiai Kiadó, Budapest, 1972.

Hock Ng Chee [1996], Queueing Modelling Fundamentals. John Wiley & Sons, 1996.

Rabiner, L.R. [1989], "*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*", Proceedings of the IEEE, 77, 2, p. 257-285, 1989.

Tijms, H.C. [1994], Stochastic Models: An Algorithmic Approach. John Wiley & Sons, 1994.

SAS/ETS [1993], SAS/ETS Software: Applications Guide 2, Econometric Modelling, Simulation, and Forecasting. Version 6, First Edition, SAS Institute Inc.

Chapter 4

Sub-Optimal Mask Search in SAPS

4.1 Introduction

This chapter introduces a new algorithm for a sub-optimal mask search in SAPS. The algorithm is not exponential as in all known previous versions of SAPS, but polynomial in nature. This results in an improved execution speed for SAPS. Moreover, it allows for handling of complex systems that were intractable before. The first part of this chapter (sections 4.2-4.4) describes the theoretical justifications and the principles behind sub-optimal mask searching.

The second part of this chapter (from section 4.5 on) deals with the implementation aspects of the sub-optimal mask search. As the sub-optimal mask search algorithm is not implemented in SAPS-II, a novel implementation in Smalltalk, called SAPS-ST (ST is the abbreviation for SmallTalk), was designed. Although the algorithm is implemented in Smalltalk, it is not intrinsically based on an object-oriented approach. The optimal mask-searching algorithm is implemented for backward compatibility reasons. Comparisons and benchmarks are thus possible because of the same uniform object-oriented environment. SAPS-ST is designed as a research tool. Incorporating new techniques for recoding and mask search is very simple. It is user-friendlier than SAPS-II and no programming skills are required in its use: even a new quality function can be interactively constructed without writing a single line of code. Finally, SAPS-ST has an option to transform an activity matrix in a form suited for the data-mining approach described in chapter 8.

Most of part 1 of this chapter is published in [Van Welden and Vansteenkiste 1996].

4.2 Optimal Mask Search in SAPS-II

As seen in chapter 2, in searching the best mask, one starts with a low order mask and gradually increases its order (depth) until the mask is found, see Figure 2.9. The search is layered with regard to the number of *m*-inputs. In the optimal mask search, an exhaustive search of all possible sub-masks is done. One starts with generating all simplest sub-masks (two entries: one input and one output) and by evaluating each. The mask with the highest quality is retained. Then one creates all sets with an extra input entry and starts evaluating all (sub)masks formed by combinations of these input entries. Again, the one with the highest quality is retained. This process continues for higher entry masks until the candidate mask is reached. The mask with the absolute best quality is then chosen to be the optimal one. This process is depicted in Figure 4.1.

The ‘take best sub-mask from a candidate set’ is elaborated in Figure 4.2.

The search has an exponential time-complexity: one obtains approximately 2^n masks to be evaluated before reaching a decision (all combinations of masks with two input entries, three

input entries, … from the candidate mask are evaluated). An improvement by some techniques to stop excessive evaluation can be built in by cutting the generation of new masks under certain observations. However, this does not change the order of complexity. Apart from this improvement, one notices that the search is exhaustive. The *optimal* mask will be found within the limits of an initial candidate mask[1]. All the others will have lower quality.

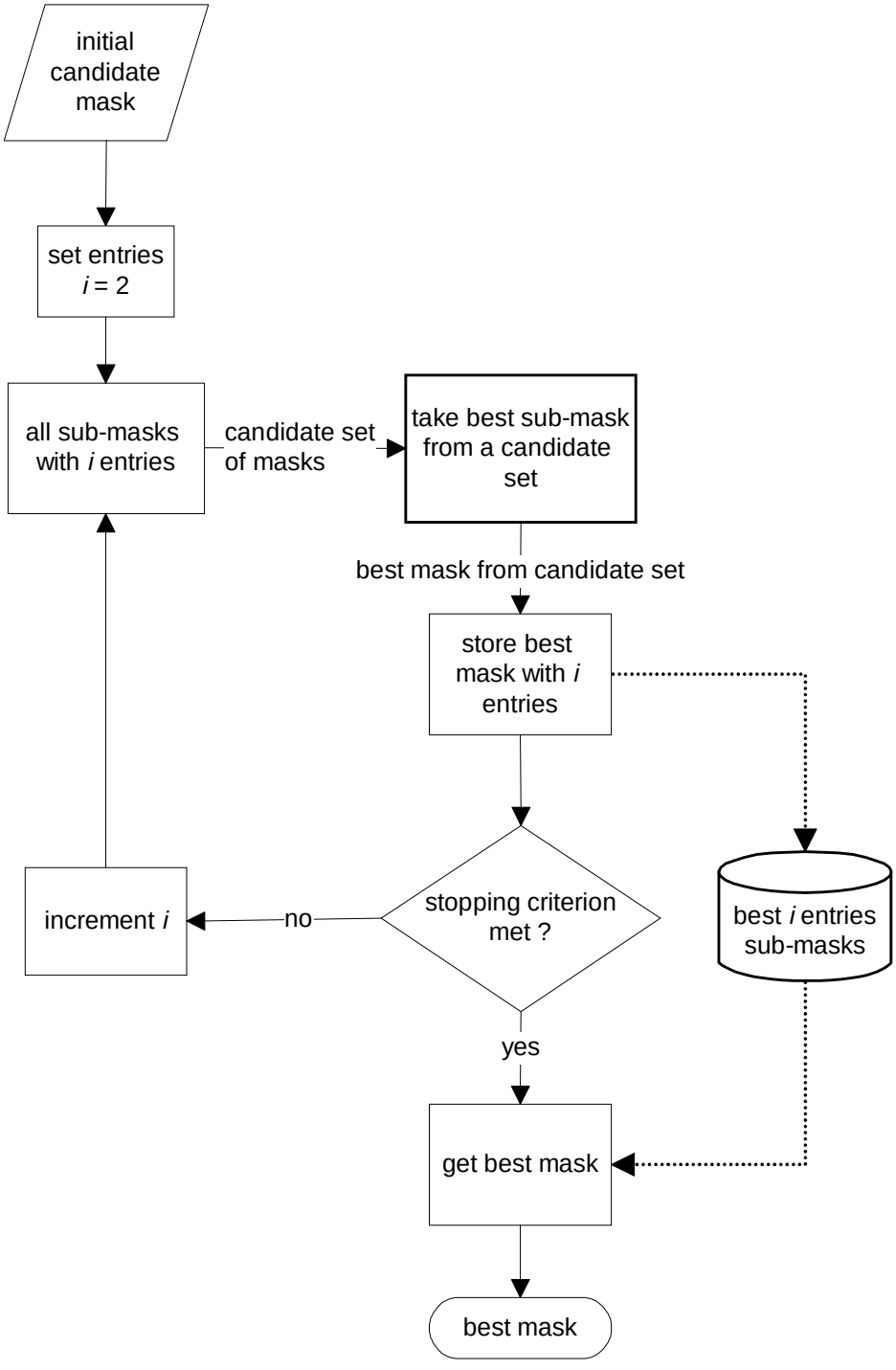


Figure 4.1 : Optimal mask search in SAPS-II

[1] The concept of an initial mask will return in the new algorithm

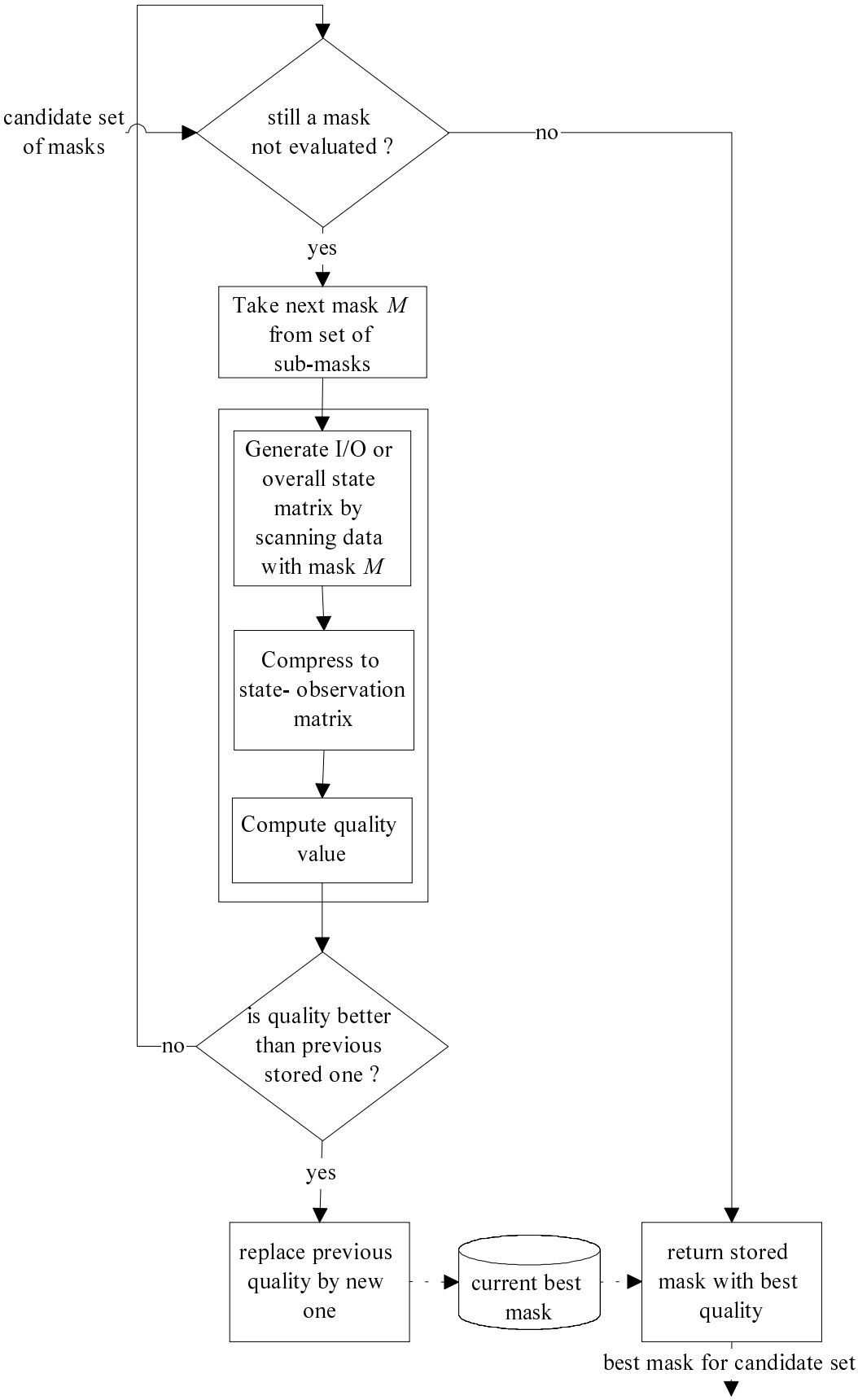


Figure 4.2 : Take best mask from a given set

Remarks

Two remarks can be made on this approach:

- the exhaustive search results in a combinatorial explosion of different masks to be evaluated. Complex systems with many variables and/or mask depths can not be analysed due to the exponential time-complexity. For systems that can be analysed, the computational cost is high for there are many masks to be evaluated.
- in the exhaustive search process, multiple peaks in mask quality are observed that differ only slightly. The maximum peak may sometimes only slightly exceed another that corresponds with a much lower complexity. Therefore, the choice — viewed by the user — of the absolute maximum sometimes may seem a bit arbitrary. An illustration is found in Figure 4.3. This figure gives the respective qualities for all masks used in an optimal mask search for the 'haunted house' example that was first introduced in [Uyttenhove 1978]. This example starts from the observation that, when tinkering with the lights and the radio in the house, some strange noises (due to ghosts) could be heard. These strange noises could be laughing, walking (on a cracking floor in the attic), or both laughing and walking. The aim of the experiment was to have quiet nights by a correct sequence of light-and/or radio switching. Hence, the inputs of the system are lights (states: on/off) and radio (states: on/off). The output was ghosts with as states: quiet, laughing, walking, or walking and laughing. After tinkering with the lights and radio, it was found that 17 observations were sufficient to detect a deterministic pattern to keep the ghosts silent.

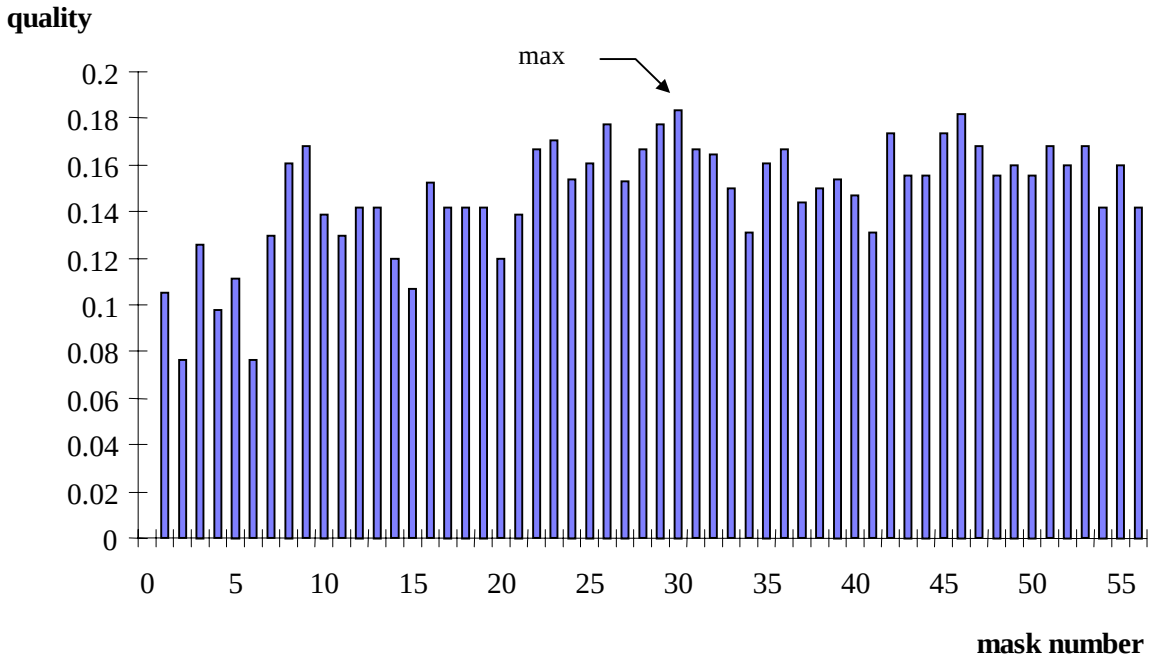


Figure 4.3 : Haunted house example

4.3 Sub-Optimal Mask Search: a new method

Due to the reasons mentioned in subsection 4.2, another approach is proposed for the searching method. Instead of starting with the simplest sub-masks in the evaluation of masks, start with the most complex one and consider this as the root of a search tree. The most complex mask of a given depth, is a mask with all *m*-inputs set to -1. Such a mask[2] is named a

[2] The reader should recall that, since chapter 2, only one-output masks are considered.

maximal allowable mask. An example of a maximal allowable mask with memory depth 2 for five input variables is given in Table 4.1. It corresponds with the most general dependency equation that goes two time steps back in time, given by

$$y(i) = \tilde{f}(u_1(i), u_1(i-1), u_1(i-2), u_2(i), u_2(i-1), u_2(i-2), u_3(i), u_3(i-1), u_3(i-2),$$
$$u_4(i), u_4(i-1), u_4(i-2), u_5(i), u_5(i-1), u_5(i-2), y(i-1), y(i-2))$$

with $\tilde{f}$ a qualitative function.

	inputs					*output*
time	u_1	u_2	u_3	u_4	u_5	y
i-2	-1	-1	-1	-1	-1	-1
i-1	-1	-1	-1	-1	-1	-1
i	-1	-1	-1	-1	-1	1

Table 4.1 : Most complex (highest cardinality) mask of memory depth 2 for five input variables

From it, a new level in the tree is generated. This is done by taking all sub-masks with one less input entry, i.e., with a lower cardinality by 1. These sub-masks are all evaluated. Only the best (sub)mask is retained for further analysis. It becomes the new parent mask for generating all its sub-masks with again one less entry. Each of these sub-masks is evaluated and again the best mask (for this level) is retained for further analysis. This process may at maximum continue until two entries are left (one is the input entry and the other is the output entry). It forms the backbone of the sub-optimal mask search approach that will be elaborated upon on in the sequel, where an alternative view on the quality function will be given and justifications for the new approach will be given.

In real and complex systems the greedy tree-traversal approach just described will not continue until leafs are met with only two entries. In order to explain this, it is important to state some formal principles about the measures attached to each node. A node contains a mask as one of its attributes. It is denoted by M_i. Other attributes will be the normalised entropy value e_i and complexity measure c_i. To search a (sub) optimal mask, one tries to minimise the normalised entropy and the complexity. This adheres to the desire to rely on highly deterministic and yet simple models. Summarising, a node n_i is determined by:

- its mask M_i,
- its normalised entropy value e_i
- its complexity value c_i (e.g. number of non-zero m-inputs, observation ratio)
- its quality q_i, (e.g. Shannon Entropy or generalised entropies)

where the quality is a function of the entropy and the complexity (see chapter 2), i.e.,

$$q_i = f(e_i, c_i)$$

The entropy measure that is considered for the quality determination is the Shannon Entropy. It is intuitively clear that the entropy will increase when one goes down the tree. After all, the more one goes deeper in the tree representation the less entries are left in the masks. Consequently, the degree of determinism governed by the equation 1-e_i, will decrease.

The complexity measure is also supposed to decrease (non-monotonically) when going deeper in the tree. As an example, take the very simple complexity measure determined solely by the number of generating entries in the mask. This complexity choice gives a level-invariant value. In this way, the measure will only promote the depth of the tree traversal by giving different weights to the levels in the tree, but it will not distinguish among different mask configurations on the same level.

The entropy increases while the complexity decreases. Thus, the quality may increase initially, go through several maxima and finally starts decreasing again. The optimum is where the trade-off between the degree of determinism and the degree of complexity is at its best, i.e., where the quality is the highest while descending the tree.

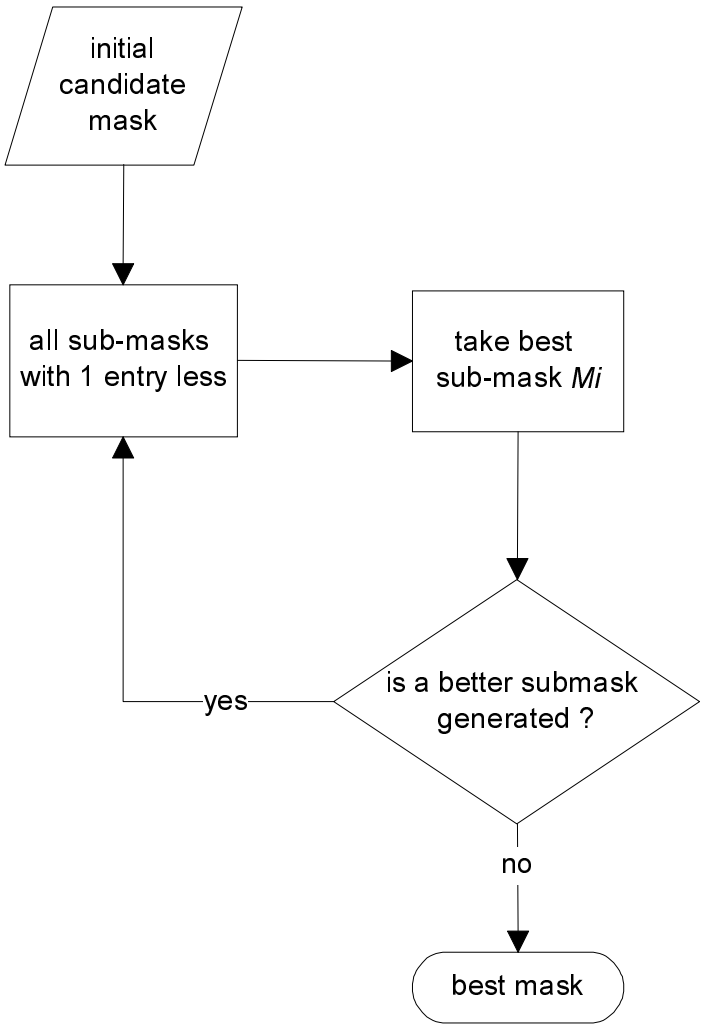


Figure 4.4 : Sub-optimal mask search

The algorithm for the (sub) optimal mask analysis is then, see also Figure 4.4:

Consider a (parent) node n_i with quality q_i.

1. check its total number of entries:
   if the number of entries is two, then stop and return M_i as the sub-optimal mask,
2. evaluate all children:
   a) if all children have a lower quality than q_i, then stop and return M_i as the sub-optimal mask,
   b) else take the first child node with the highest quality, consider this child as the new parent node and
3. repeat the process from step 1.

Obviously, the quality function is used as an evaluation function.

In case (1), the search process is stopped because the node is a leaf. For if there are two entries left (one serves as input, the other as output), no further reduction is possible.

In case (2a), the search process is stopped because the quality begins to decrease. The compromise is thus reached at the current level and no further search is considered necessary. All children nodes have a lower quality because the increase in entropy outweighs the decrease in complexity.

In case (2b), the search continues because the compromise has not been reached yet: still simpler (sub) masks fulfil the criteria set up. Some children have a higher quality because the increase in entropy is overshadowed by the decrease in complexity.

Looking at the algorithm, one notices that the sub-optimal mask search mechanism is conducted by a hill-climbing strategy [Winston 1992]. All children of a current node are evaluated and only the best child is selected for further expansion. Consequently, an optimisation is done level per level in the tree (one could talk about local optimisation in that aspect). The search halts when all child nodes have a lower quality than their parent does. The parent is then the sub-optimal mask.

The hill-climbing strategy is a greedy approach, and it is based on heuristics. In this way, one can circumvent the computational complexity of the search problem, but without guaranteeing to have found the optimal mask. Objections against the latter argument will be considered in section 4.4.1, where a justification for the sub-optimality of the search will be given.

The tree structure is clearly visible in Figure 4.5. The root of the tree is the maximal computationally allowable mask (here with memory depth 2). The maximal allowable mask is set by the user. Usually, all state variables are included in the mask, but the maximal memory depth of this mask depends on implementation parameters (such as the programming environment or language, the type of computer with its amount of RAM, disk space, parallellisation, etc.). Sub-nodes are formed by putting an entry zero. In fact, the tree is not really a tree because of the redundancy of the different sub-masks (i.e., it is a lattice), but for searching purposes (hill-climbing), one may consider it as a tree. Remark the ‘top-down’ approach as contrasted with the chosen ‘bottom-up’ approach of the exhaustive search.

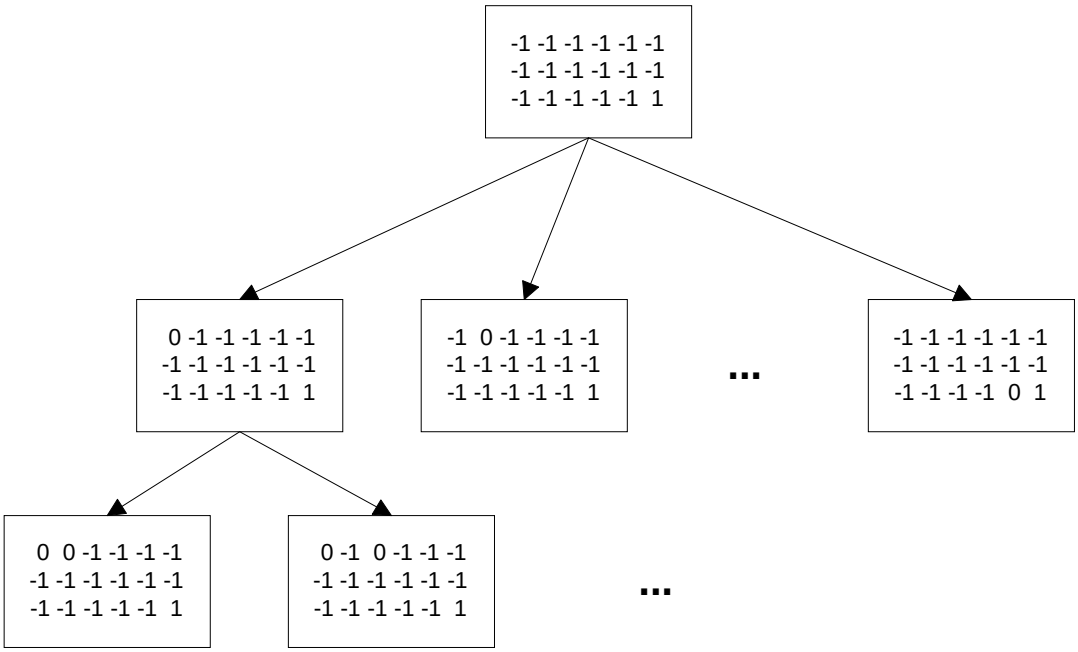


Figure 4.5 : Tree decomposition for the maximal allowable mask in Table 4.1

Because of the top-down approach, the maximal allowable mask has also a greater significance than in the exhaustive search. It not only serves as a computational constraint (which is its function in SAPS-II), but it also serves as a starting point for the mask search (root of the tree).

Contrary to the exhaustive search a specific search structure is employed in the sub-optimal approach.

4.4 Comparison of optimal and sub-optimal mask analysis

It is crucial to realise that the quality measure is in fact a kind of evaluation function, which has to be based on a (hopefully) sound heuristic (the use of a heuristic will re-appear for tree classifiers in chapter 6). Considering it this way, it is advisable to set up the state space for searching the 'best' mask as a tree structure where the candidate mask for the system is located at the root. In the optimal mask analysis, the search is done bottom-up 'breadth-first', i.e., level by level towards the candidate mask while evaluating and retaining the best mask for each level. In the sub-optimal search algorithm, the search is conducted top-down; away from the candidate mask.

Unfortunately, heuristics are fallible. The heuristic used here is nothing more than an intelligent guess of what should be the next mask to be evaluated. The heuristic cannot foresee what will be the quality values further down in the tree and for the descendants of the sibling nodes. Thus, the heuristic can lead to a sub-optimal mask that is different from the optimal one, which is found by an exhaustive search. The sub-optimal masks correspond with the near maxima[3] in the quality curve (see Figure 4.3 and Figure 4.6[4]).

---

[3] Using the term 'local' maxima can be confusing as there is no order relation in the mask generation index. With the term 'near' maxima, it is meant that the found sub-optimal masks can be identified with peaks nearly as high compared with the highest peak in the quality 'trajectory'.

[4] Figures 4.6, 4.7 and 4.8 stem from the example in appendix A.

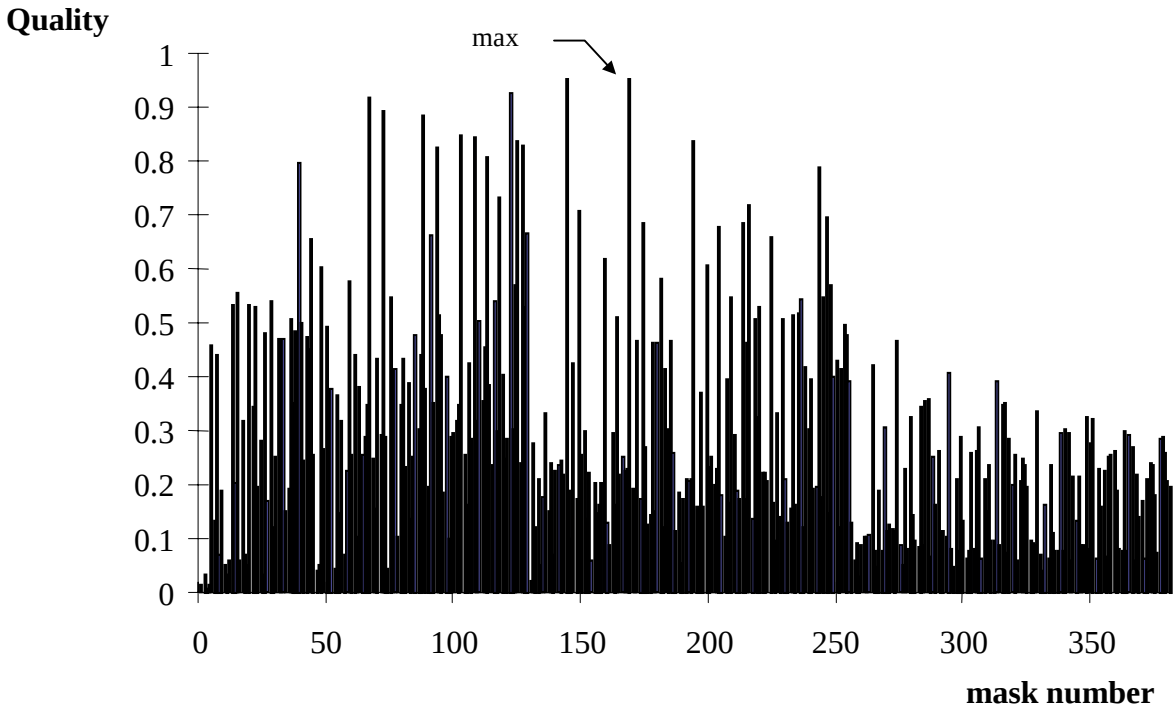


Figure 4.6 : A linear system : first output

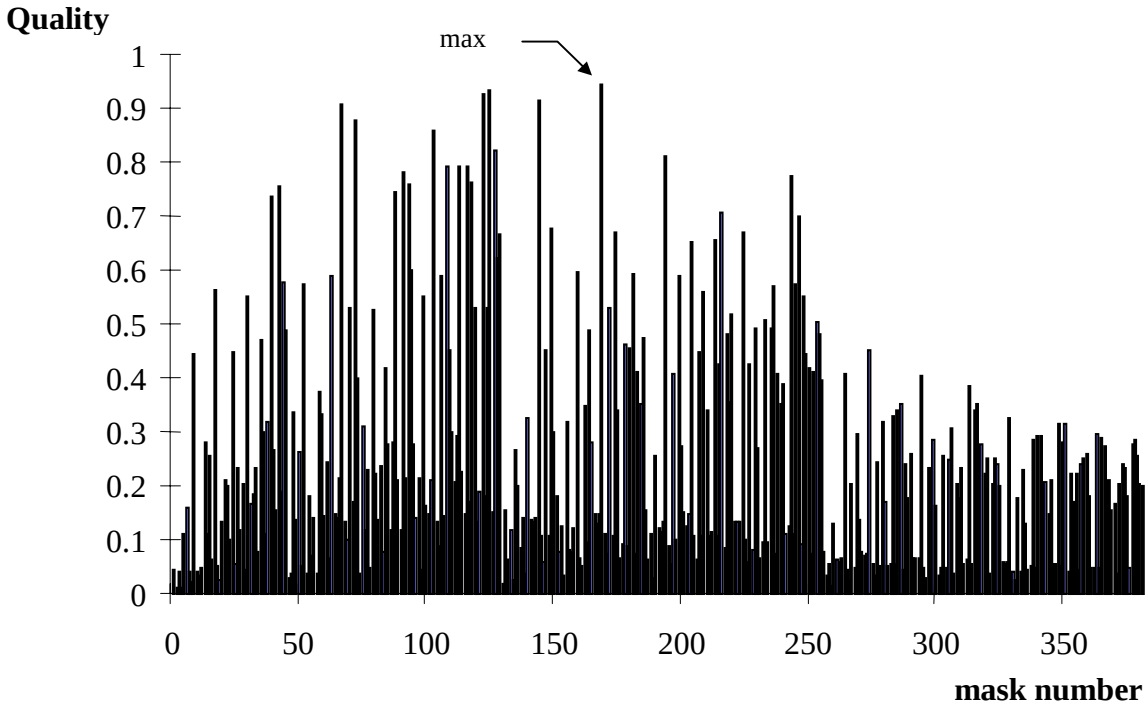


Figure 4.7 : A linear system : second output

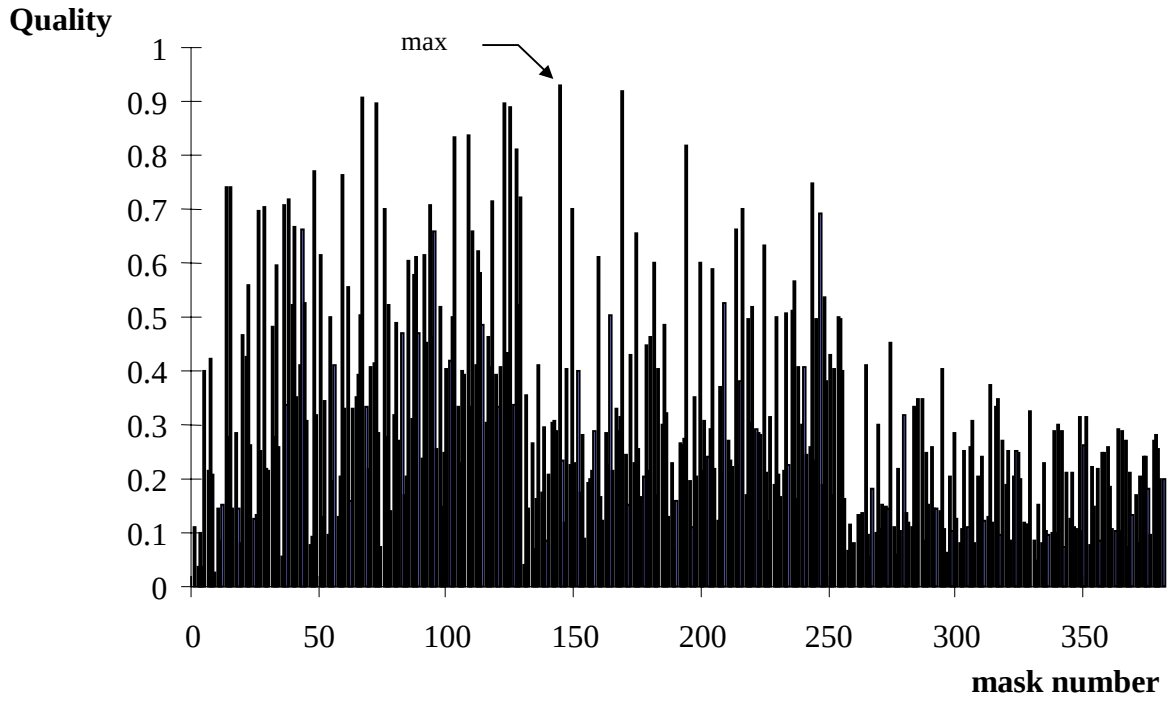


Figure 4.8 : A linear system : third output

Note: All figures depict the quality as a function of the mask number. The mask number increases from left to right. The simplest masks have low mask numbers. The candidate mask corresponds with the highest mask number. An arrow indicates the optimum found by SAPS.

4.4.1 Sub-Optimal Search; is it justifiable?

One may ask if it is worthwhile and justifiable to use the sub-optimal mask search. In the following, some reasons why this is the case, are given.

A mask doesn't necessarily represent a physical underlying principle.

The masks tested out in the search are no products of physical underlying principles. They are just possible explanations for patterns found in the data. They are very concise means to express time-invariant relationships in the data. No meaning is attached to a mask, anyway, not a-priori. Of course, every physical system has physical grounds on which it is based. Therefore, a mask for that system should preferably stand in relation with the physical properties of the system. Nevertheless, this is not necessarily the case. For example, SAPS can be tried out in economical systems. There, most often, one does not have even the slightest idea what principles are responsible for the observed behaviour because of the high degree of interactions between all the variables. Finding a pattern that allows to do forecasting is already a big step towards the simulation of such systems.

Often, the optimum found is not significantly higher than some other points.

There is not always a practical reason to prefer one and only one optimal mask as the best. In analysing data, it is discovered that the envelopes of many quality curves are relatively flat. Peaks are superimposed on this envelope and the optimal mask algorithm only selects the highest peak. However, if one looks at the other peaks, then some are found that are nearly as high as the maximum but with a much lower complexity. Their corresponding masks could have been selected by a human, who prefers their higher degree of simplicity and who does not care for the very small relative deviation from the optimal mask. Therefore, the optimal mask analysis uses brute non-intelligent force to find an optimal mask. It is not asserted that

the sub-optimal mask analysis finds a ‘better’ mask; it is only claimed that the optimum mask may not be so optimal. Consequently, the use of a less expensive search technique for sub-optimal results may be justified. This is illustrated in Figure 4.3, in Figure 4.6, Figure 4.7, and Figure 4.8. Figure 4.6, Figure 4.7, and Figure 4.8 are determined with SAPS-II. The data stems from the example of a linear system described in [Cellier 1991].

The optimum point is very sensitive to the quality of the data, the measures taken and the quality function.

There is also no exact solution due to the uncertainty in every step of the data processing in SAPS, so the use of heuristics may be motivated. Although one may select an absolute maximum in quality and take the corresponding mask as the optimal one, there is no guarantee at all that it is the best. As in the previous item, this is not about the technical issue of neglecting small differences. The main reason here lies in the lack of knowledge about the system. Even in the recoding of the data, many different ways for doing that are possible and each way may result in another optimal mask. In the mask analysis itself, the choice of the measures is very important. There is no objective and uniform right way to choose a specific set of measures for different kinds of entropy measures and complexity measures are possible and may be as justifiable as another). This poses some severe problems. A good quality measure can only be determined empirically. The effect of the choice is important for both algorithms.

It may seem that the implication of a certain choice of quality function with its arguments (measures) is greater in the sub-optimal search. However, if one really takes only the maximum quality mask, then a small change in quality determination may change the optimal mask radically and the previous found optimal mask may be lost. The optimum mask is thus ‘context-dependent’.

In the hill-climbing search, no recovery can be done. Hill-climbing may become stuck at a local maximum (or minimum). If the search halts, then no consideration is made anymore for the grandchildren and no attention is ever been paid for the siblings (and their descendants). Maxima present in these nodes will never be detected. Another drawback of hill-climbing is that it can get stuck in infinite loops. Fortunately, this can never be the case here, because the state space is a directed tree.

The optimal mask may be intractable.

Even under the assumption that the optimal mask would be the ‘best’, technically and epistemologically, the computational cost of finding one may be prohibitive. The number of masks to be considered in an exhaustive search is too high in complex systems (exponential). For both algorithms, one can easily calculate the number of masks to be evaluated under a given maximal allowable mask, by noting that (n is the number of m-inputs of the maximal allowable mask M, i.e., $n = \#M - 1$):

- for optimal mask search, there are:

$$\binom{n}{1}+\binom{n}{2}+\binom{n}{3}+\cdots\binom{n}{n} \tag{4.1}$$

  (one-output) masks to be evaluated and,

- for sub-optimal mask analysis with a hill climbing search algorithm this number becomes:

$$\binom{n}{n-1}+\binom{n-1}{n-2}+\cdots+\binom{3}{2}+\binom{2}{1}+1 \tag{4.2}$$

Thus, in an optimal mask search there are approximately 2^n (one-output) masks to be considered, whereas in the sub-optimal case, there are 'only' approximately n^2 (one-output) masks to be considered. The exponential time-complexity is reduced to a polynomial of the second degree. For complex systems, this may make the difference between an intractable system and a tractable one. For relatively simply systems, an improved execution speed is the mere result.

The sub-optimal mask algorithm allows larger candidate masks to be tested. This has an implication on the number of variables that can be considered and on the time constants of the system that can be captured. In [Cellier 1991], one finds that the largest time constant that can be captured is usually limited to three or four. This can now be relaxed to a higher number.

Even in SAPS-II, less optimal masks are used to forecast data.

This argument is a bit opportunistic for it is based on experiences with SAPS-II. When an optimal mask is found in SAPS-II, it is possible that it doesn't contain enough predictiveness to allow a complete forecasting. Cellier coped with this by selecting the three best masks, where the second has a lower complexity and the third an even lower complexity. The second and the third mask are not optimal, but they may be indispensable for the forecasting. Hence, different kinds of sub-optimal models are used.

The optimal search strategy is not similar to the human way of searching a pattern.

When humans solve a problem, they do not evaluate all possible combinations of patterns but use heuristics to guide them in the selection of a relevant pattern. Although, in this case a non-exhaustive heuristic-based bottom-up way of evaluating masks is perhaps closer to their way of looking at patterns.

In the next section, the performance of the two mask algorithms is compared for three examples. The first is simple and is called the haunted house example. It is taken because it was initially used to verify the new SAPS-ST version with SAPS-II. The second illustrates the real power of the sub-optimal mask algorithm. It is a more complex example, which stems from the medical domain. The third originates from appendix B.

4.4.2 Examples

The graph for the haunted house example that is depicted in Figure 4.3 shows that the curves are quite flat and that a global maximum is not that better than the second best, third best, etc. It shows that more simple masks may also be quite good for a given purpose (compare this with the philosophical differences between machine learning and statistics about accuracy and comprehensibility of a model in chapter 5).

The real impetus for finding a new search algorithm came from another example. A physician was looking for a pattern in medical data [Barreto 1994]. The data consisted of 19 classes of symptoms (input variables) for finding the disease (output variable). These 19 symptom classes may be redundant, but this not sure. Each symptom was already classified according to the qualitative terms like: 'high', 'medium', 'low', etc. by the physician. There were three possibilities for the disease underlying the symptoms (three qualitative values). The aim of the doctor was to discover the simplest relevant pattern between the symptoms and the disease so that not always all 19 symptoms had to be investigated (some of them required extra investigations, cost and time)[5]. Trying to use the two versions of SAPS with the optimal search algorithm did not work. This example shows that the optimal mask synthesis fails. In

[5] This problem is a prototypical example for a technique described in chapter 7.

this case, there are simply too many variables. The sub-optimal mask search algorithm, however, finds a mask quite easily. It demonstrates the real necessity of a sub-optimal search algorithm if the system is too complex.

The relative importance of the best mask is also illustrated by Appendix B section B.3, although this appendix serves other purposes that are explained in chapter 8. The data is generated in Matlab, [MathWorks 1999], and consists of a sine wave contaminated (added) with random noise. The resulting signal is delivered to a switch that is controlled by a block pulse generator. For this example, an optimal mask was sought. The optimal mask showed a dependency of the form (quality is 0.9387):

$$out(i) = \tilde{f}(\text{out}(i-10), \text{sine}(i-9), \text{pulse}(i-7)),$$

while the second best showed a dependency of the form (quality is 0.9373):

$$out(i) = \tilde{f}(\text{out}(i-10), \text{sine}(i-9))$$

The quality of the latter is slightly less than the former, but looking at the state-observation matrices show that they do not differ much with regard their variability in prediction (see appendix B). Performing a prediction on a test set shows almost a same performance with regard to accuracy, while the latter is a simpler model. Table 4.2, which shows the Sum of Squared Errors (SSE) and the Sum of Absolute Errors (SAE) on a validation set, confirms these findings, see also Figure B.4 in appendix B.

	SSE	SAE
Optimal mask	202.4	101.7
Second best mask	199.7	100.8

Table 4.2 : Sum of Squared Errors and Sum of Absolute Errors

The forecasting method is the same for both masks and is done on the same data set, so it makes sense to compare them. The least one can say is that there is no rigid justification for preferring the best mask to the second best in this case, which is in concordance with the statements made earlier about 'optimality' of a mask.

4.5 SAPS-ST: a prototype tool for GSPS

The sub-optimal algorithm dealt with in section 4.3 is implemented in VisualWorks 3.0©, [ObjectShare 1999], which is a Smalltalk (object-oriented) development environment. The SAPS-ST system is composed of independent units: a recoding module, a pattern analysis module, and a forecasting module. In the pattern analysis section, a natural mapping from the epistemological levels of GSPS to a class hierarchy in Smalltalk is established. In a first version of SAPS-ST the source-system, the data system and the behaviour system were implemented in a straightforward manner, based on the knowledge inheritance principle of the epistemological levels, [Van Welden and Vansteenkiste 1994]. The new version is more pragmatic in the sense that the object system is removed[6], and that recoding is extended. Additionally, for the purpose of data mining (see chapter 8), mask shifting over raw data is now possible. Besides this, the question to whether recoding belongs to the object or to the

[6] In former versions variables were set a priori and it was checked if the data complied with them. Now the starting point is an activity matrix from which the variables are defined. I did not try to automate the data-gathering process anymore.

data system cannot be resolved fully. Therefore, the new implementation is now entirely data-driven: it starts from the reading of raw data. It is assumed that these data contain at least the relevant variables, and that pre-processing is done already to eliminate trivial errors (see data mining: pre-processing the data). It is also assumed that the sampling occurred at the proper time-instances. Figure 4.9 shows the structure of the SAPS-ST program.

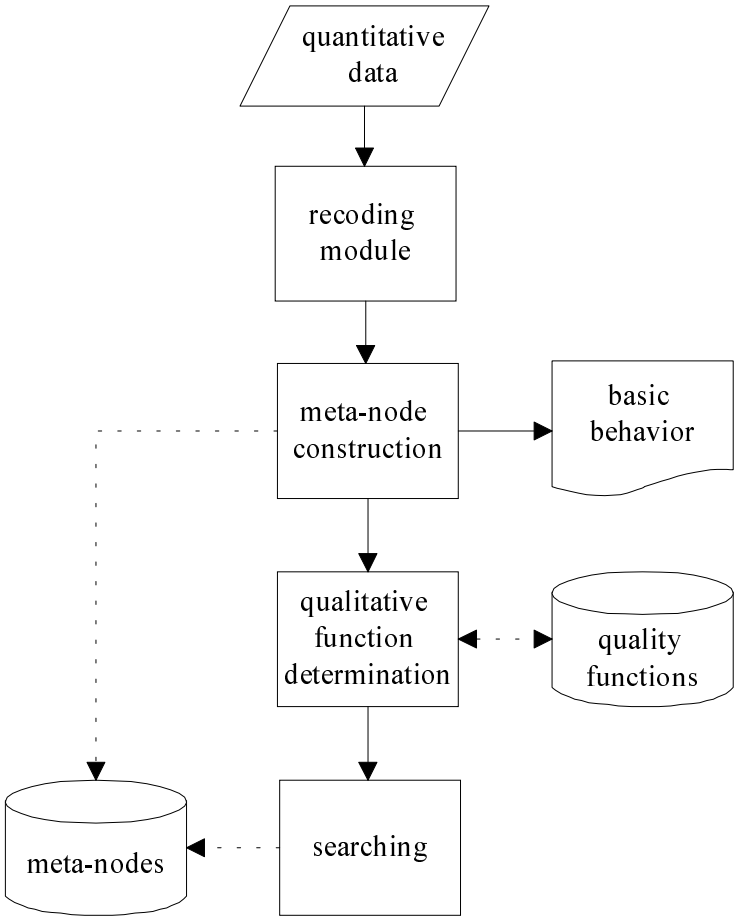


Figure 4.9 : Structure of SAPS-ST

A training set of data is read from a file and recoded. This is the SAPS-ST implementation of the 'data' system. An extra extension in the newest version of SAPS-ST is that more recodings for a variable can be generated and used in the subsequent process of mask searching (all recodings are present in one mask). Once data is read in, primary masks can be set forward. A primary mask stands for a maximal allowable mask as depicted in Figure 4.5. Different primary masks can be constructed to represent different outputs, to allow different search methods, or simply to look at the effect of different maximal allowable masks for certain search methods. Each such primary mask is stored in a meta-node. This is a Smalltalk object with the possibility to generate sub-nodes, each containing a sub-mask and a corresponding state-observation matrix. Thus, a meta-node knows how to evaluate a mask and how to create a node for that mask. The searching among different nodes is done in a separate module that is responsible for the mask-searching. The exhaustive search and the sub optimal search form two primary methods in the inference module. More elaborated searches can be plugged in easily. Although the module may change drastically during further development, the modification will not affect the pure calculation part where the criteria are boiled down to calculable measures. This is due to the object-oriented approach and a clearly defined interface between inference module and quality computation module (the meta-node). The

searching module requests to a meta-node a set of sub nodes, and it solicits the meta-node the quality for each node. Then, the searching module determines what to do next: what new nodes should be generated, etc. Thus, the searching module works with nodes and their respective qualities, but it does not look into the computation of this quality in a node. The latter is the responsibility of the node itself (via its meta-node). The determination of the quality function can be changed from meta-node to meta-node. The object oriented programming style also allows for very flexible insertion of new criteria for evaluating masks, which can be tested against already existing evaluation criteria and algorithms. This can be changed at will without influencing the search (inference) mechanisms. Therefore, downward compatibility is maintained in the same environment. This opens the possibility to look at the effect of different quality functions on a certain search strategy. Hence, the system is very versatile in exploration of different quality determinations, search techniques, mask constructions, recoding techniques, etc.

The main interaction window in SAPS-ST is shown in Figure 4.10. The top pane represents the data system, the middle pane allows for mask construction and for the (sub) optimal searching process. It also gives the possibility for forecasting. In that pane, different primary masks can be built and for each mask in the list, a separate search for the 'optimal' mask can be executed. Hence, different masks and/or searching techniques can be compared and evaluated. The bottom pane is an output pane. All what is shown there can also be dumped in a file for later inspection or for sharing data results with other applications.

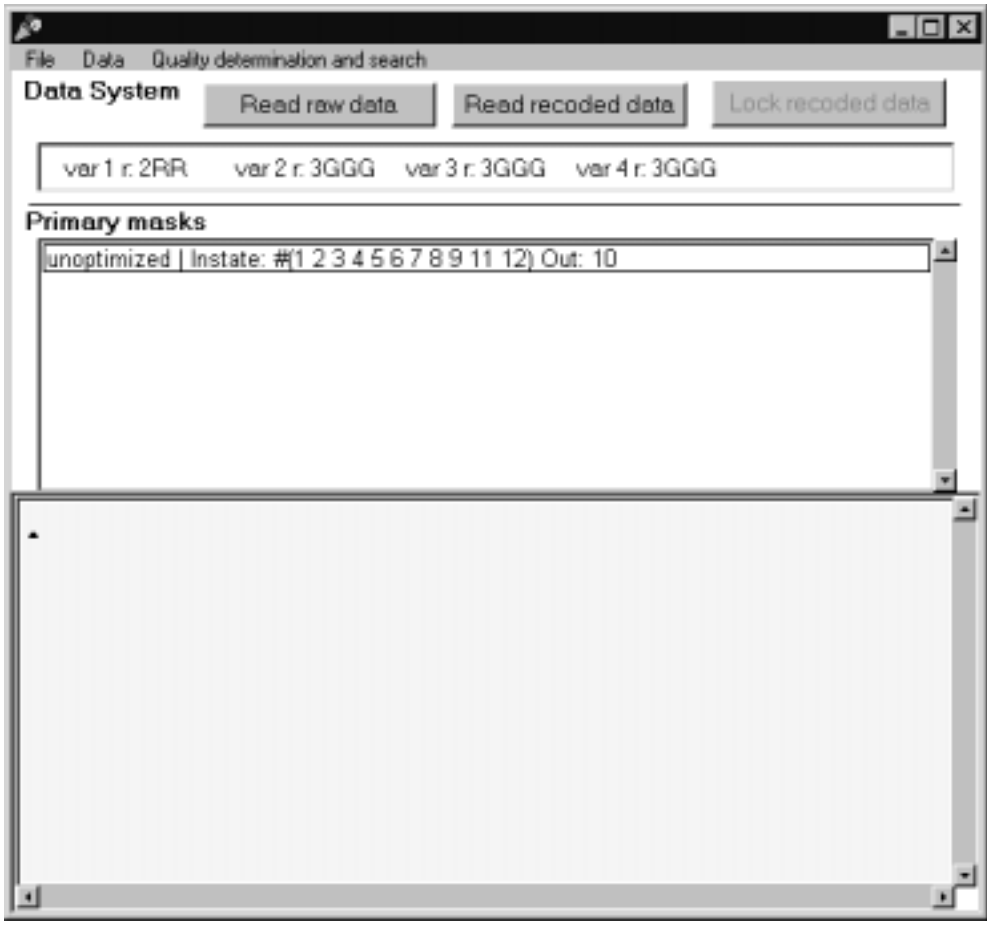


Figure 4.10 : Main SAPS-ST interaction window

4.6 Recoding in SAPS-ST

The recoding in SAPS-ST is similar to the recoding used in SAPS-II, but it has more possibilities, and it has a user-friendlier interface. The latter makes recoding in SAPS-ST fully interactive and very easy to do. It consists of making choices, clicking on buttons, and letting the system work it out. Hence, the user does not need any programming experience anymore.

After reading the raw data, a window with all the read-in variables is opened and the user can select each variable in turn to recode. This is depicted in Figure 4.11, where four variables are ready to be recoded. In the activity matrix, the columns representing the variables are labelled to maintain a better view of what happens to what variable. An entry in this matrix can be any data-type (e.g. numbers, strings or characters). The system automatically recognises if the variables are valid numbers or if they stand for nominal variables. This enables working with textual identifiers like ‘low’, ‘medium’, ‘high’, etc. right away

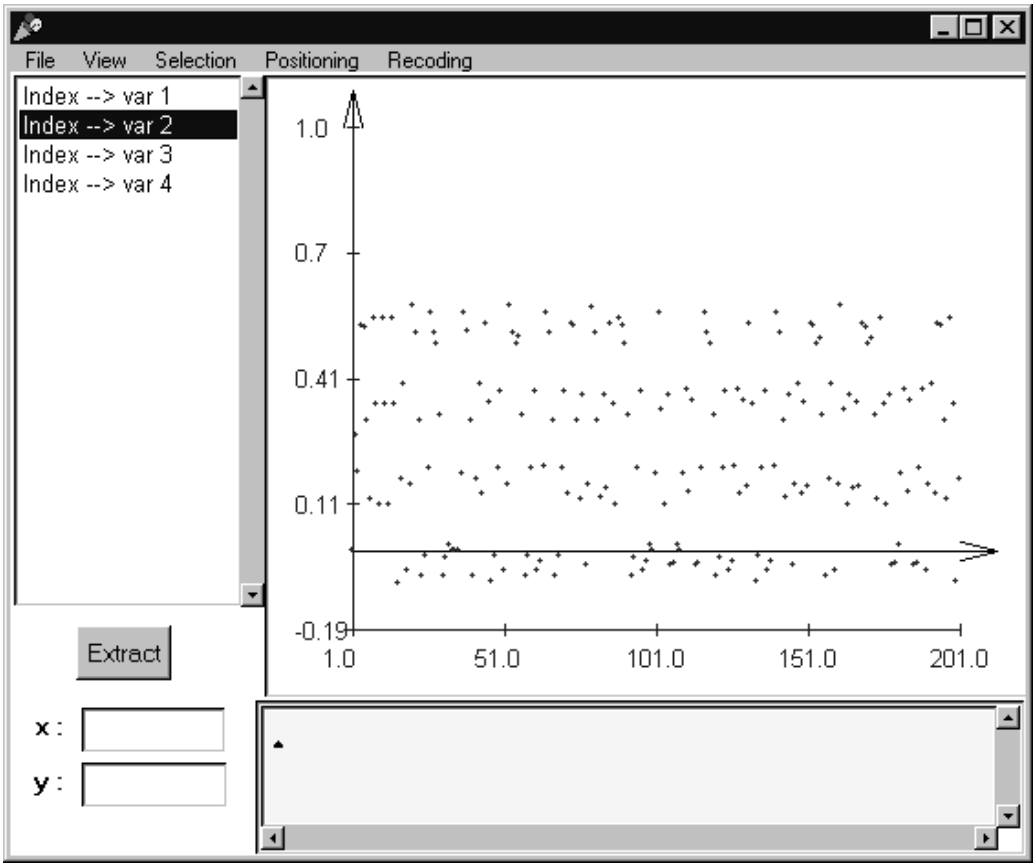


Figure 4.11 : The data window for recoding

The recoding window for a selected variable that is taken to be recoded from Figure 4.11, is shown in Figure 4.12.

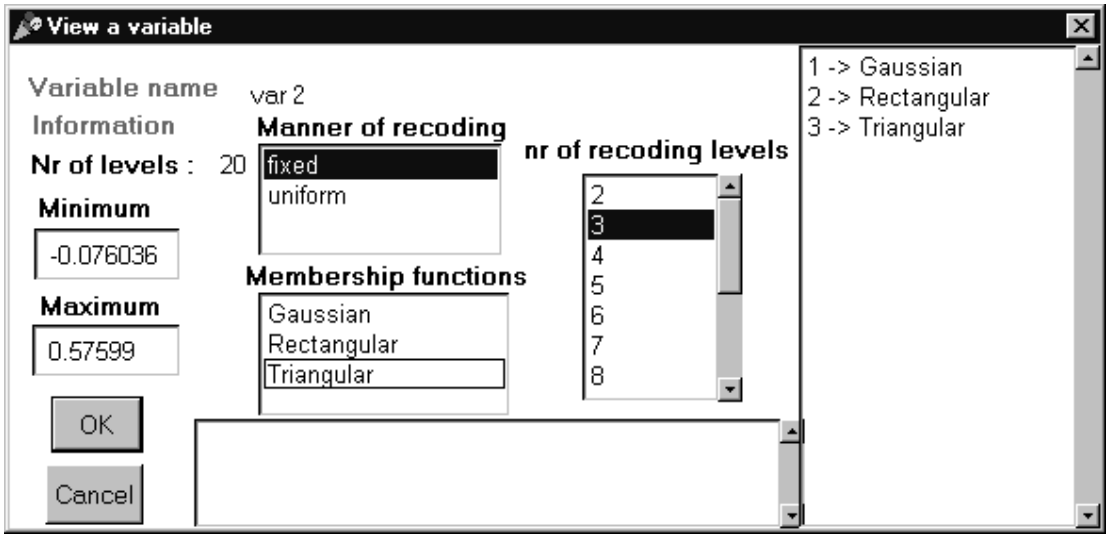


Figure 4.12 : Recoding a variable

The recoding interface itself is fully window oriented. The interface is very easy to use and restricts the user's actions to eliminate faulty user commands. The determination of landmarks in the quantisation of the variables can be done according to equidistant intervals (fixed-sized recoding), or by trying to keep the distribution of field values over the levels as uniform as possible (uniform recoding). An enhancement of SAPS-ST is that one can also recode manually. Figure 4.11 suggests indeed that putting landmarks in the horizontal oriented 'gaps' between the data clouds may give a good recoding. A data-dependent choice is given by the system for the choice of the number of recoding levels. One can use crisp and/or fuzzy recoding. Crisp recoding (Rectangular membership function) is considered as a special case of fuzzy recoding. For each variable, one can recode Gaussian, Rectangular, or Triangular (new in SAPS-ST). Other membership function choices can be added without much trouble due to the object-oriented set-up of the software. Just add a new item name in an array, write the corresponding subroutine (only a few lines), and it works. A further enhancement is shown by Figure 4.12: one can recode differently (with another membership function) for each interval (level) for a given variable. This is another illustration of the purpose of SAPS-ST as a prototype tool.

After recoding a variable, the result can be inspected in a data window that shows the raw values, the recoded values, and the membership functions (the user can select what he/she wants to see). Different recodings can be applied for one variable and stored in a recoded activity matrix for further processing in the mask analysis step.

4.7 Evaluating and searching a (sub)optimal mask in SAPS-ST

The first step in mask analysis is to construct the primary or maximal allowable mask. For this purpose, one selects from the set of variables the one that is the output, and the ones that are the inputs for the system. Figure 4.13 shows this process. Initially, the left list contains all the variables. By clicking arrows, one classifies a variable as system input or as system output. The system only accepts one system output for a given primary mask. If one wants to define other outputs, one has to add another mask for it. Variables that are not classified as system inputs or as system output, are considered irrelevant for the dependency relation. In other words: they will have all zeros in the mask. In the mask construction window, one finally selects the memory depth of the mask.

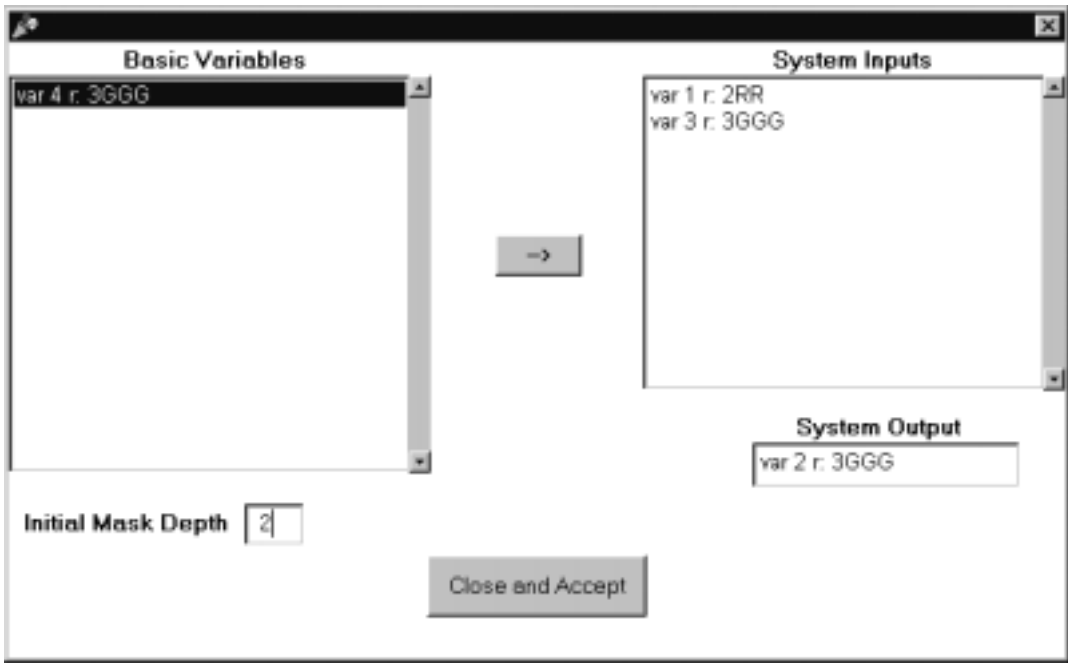


Figure 4.13 : Construction of a mask

After this basic construction, one can still edit individually entries in the mask from the main window (from Figure 4.10).

4.7.1 Extra measures introduced in SAPS-ST

In SAPS-II, the entropy is calculated by using confidence as an estimator of the conditional probability, see equation (2.8). In SAPS-ST, this is denoted by 'ShannonEntropyConf'. In SAPS-ST, the input probabilities can also be determined by taking the ratio of number of occurrences of the input state to the total number of states. This is an estimated probability based on relative frequencies, i.e.,

$$p_i = \frac{n_i}{\sum_{i=1}^{m} n_i}$$

Changing the approximation towards calculation of the input probabilities may result in large differences in mask qualities. These may then give an impact on the choice of an optimal mask. It illustrates the sensitivity of the search algorithm to the choice of measures. One can indeed ponder about the relativity of the value of an optimal mask (see 4.4.1). It also demonstrates the necessity to explore further the influence of different measure choices and the need for a sensitivity analysis. This could be a major research topic in SAPS-ST. The Shannon entropy where the input probabilities are based on relative frequencies is denoted by 'ShannonEntropyProb'.

As seen in equation (2.10), the Shannon entropy is normalised by dividing it by H_{max}. In SAPS-II, H_{max} is computed by $H_{\max} = \log_2 n$. SAPS-ST supports this too under the name of 'worstEntropySimple'. Two variants are defined in SAPS-ST, which both rely on the found state-observation matrix. Referring to equation (2.6), one could also compute a worst entropy for n_i output levels for input state i by $H_i = \log_2 n_i$. What is called in SAPS-ST worstEntropyNonWeighted, is the maximum of all these worst input state related entropies, i.e.,

$$H_{\max,non-weighted} = \max_i H_i = \max_i [\log_2 n_i].$$

If a certain input state i gives rise to all possible output levels, then $n_i = n$, and $H_{\max,non-weighted} = H_{\max}$. The measure 'worstEntropyWeighted' is a weighted version obtained via the mean value of all H_i (or expectation value), i.e.,

$$H_{\max,weighted} = E[H_i].$$

Until now, no profound research has been done to evaluate the effect of these new definitions, but is has been observed that they do have an effect on the found optimal mask (e.g., another one is found when changing the normalisation factor). Extensions in SAPS-ST, which are in the form of additional measures for under-determination (incompleteness) of qualitative data, are defined and used in the quality expression. Under-determination stems from the principle that if there is not enough variance in the data the system will never be able to discover some relevant features in the data. This can be illustrated with an electronic flip-flop device. If one generates data where the reset input port is always kept on zero, then the fact that the reset input can reset (put out on zero) the output will never be seen in the data. The latter will never be discovered. The resulting incompleteness can only be detected under special conditions where one knows the input range of every input signal. In that case, one can use a measure to determine the degree of completeness of the qualitative data for the corresponding source system (the case where relevant variables are not detected although they play a role is a matter of the source system). The incompleteness of the qualitative data has a lot to do with the forecasting power of the model. To take the flip-flop example again: one will never be able to predict what will happen if the reset input is triggered, because this behaviour never has been seen in the data before. Thus, the predictiveness, which is a measure of the forecasting power, of the model will be bad for this case. As a rule one can say:

The more under-determined the data the less predictive power there is.

The terms under-determination and incompleteness are closely linked to the concept of expressiveness. The expressiveness of a model is a measure for the information content or resolution that the model provides. The more expressiveness one wants to put in the model the more levels one has to retain in the recoding. However, the more levels are kept for a given data set, the more missing combinations will occur in the qualitative data set and the more incomplete the qualitative data will be. More expressiveness gives a fine granularity in prediction, but a lower forecasting power.

Expressiveness and predictiveness are competing measures.

Consequently, a trade-off has to be found. Cellier [1987] takes three to five levels (an odd number is preferred). SAPS-ST is designed not to be so strict. It only puts a restriction on the number of levels consistent with the data by guaranteeing a sufficient number of points for each interval (at least 5). The latter principle has led Cellier to the definition of an observation ratio as defined in equation (2.24) and a corresponding quality expression in equation (2.25) in chapter 2. The observation ratio is denoted by 'observationRatio'.

Other measures related with predictiveness are:

- 'dataRichness', which calculates the number of possible inputs in a state-observation matrix, it is the number of legal states, formerly denoted by n_{leg}.
- 'predictiveness', which is the ratio of the actual found number of possible input state found in the state-observation matrix to the possible ones. It gives an estimate of the probability that a random input state can be predicted.
- 'stability', which is based on entropy of input probabilities. It returns the entropy of the input states found.

Finally, two measures for complexity can be used in composing quality expressions:

- 'maskDepth', which is the memory depth of the mask, i.e., ΔM - 1
- 'nrOfStateVar', which is the number of state variables, i.e., the number of non-zero entries in the mask,

4.7.2 Enhancements with regard to quality function determination

The previous series enhancements concern the introduction of many new measures that can be used in the quality function. The second enhancement is that the user can freely use all these components and compose his/her own quality function description. The latter happens completely via drag and drop techniques. Figure 4.14 shows the realisation in SAPS-ST. The top pane has the quality functions that are already defined. The user can choose one of these expressions for each mask search. The pane below is where new quality functions are constructed (it is empty in the figure). Construction of a quality expression happens by dragging composing measures from the left bottom pane and/or operators from the right bottom pane to the middle pane.

These enhancements clearly show the role of SAPS-ST as a research tool.

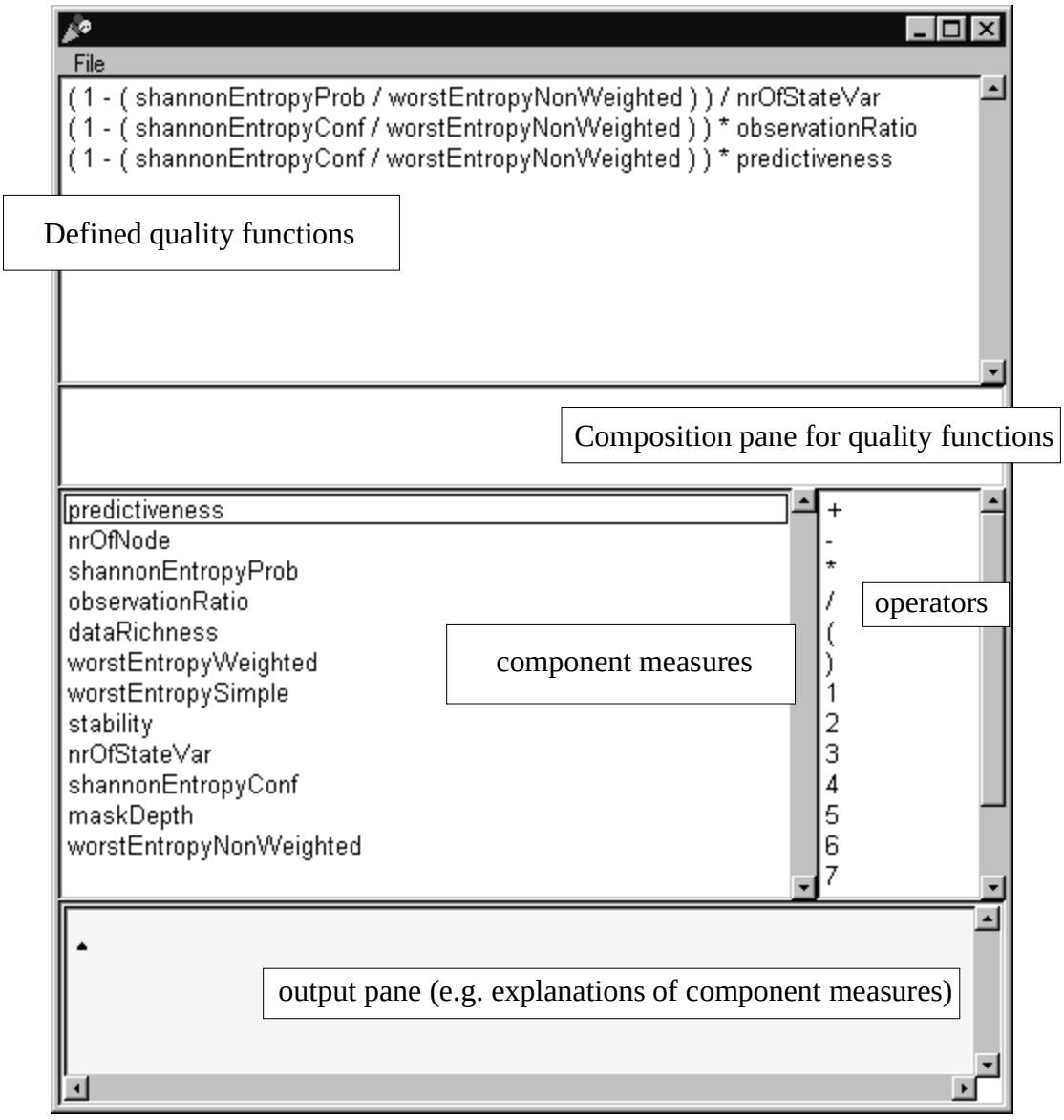


Figure 4.14 : Composition or selection of a quality function

Experiments showed that, for the same choice of quality function and search algorithm, SAPS-ST is able to mimic the output of SAPS-II [Van Welden 1999]. Section 4.4 showed that the determination of the quality of a mask is quite subjective. Hence, in SAPS-ST modifications and extension were tried/added in two ways: in the search method and in the determination of quality (orthogonal or independent extensions).

4.8 Forecasting in SAPS-ST

SAPS-ST has the ability to forecast with a given mask. Usually, one uses the 'optimal' mask, but any mask can be tried, see Figure 4.15. The quantitative data that forms the validation data is recoded just as the training data were (they can be two data sets or the same). A mask is picked from the database of meta-nodes and forecasting can take place.

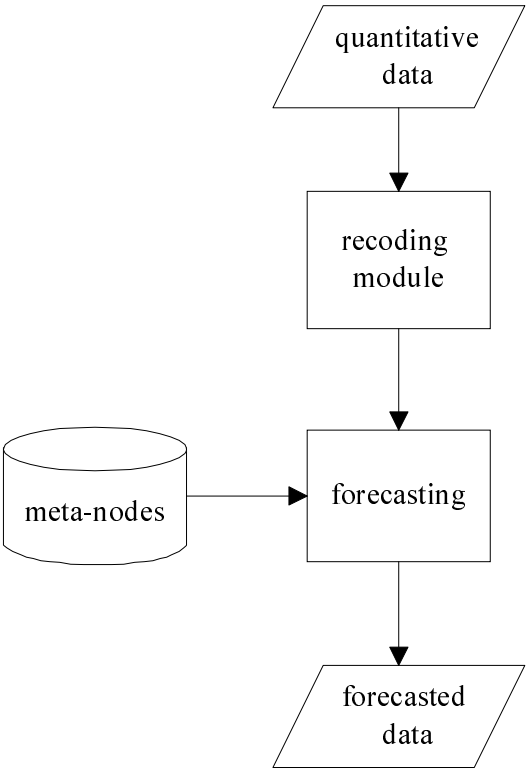


Figure 4.15 : Structure of forecasting

The two forecasting methods described in chapter 2 (but without prediction error determination for nearest neighbours) are implemented in SAPS-ST. Both methods can be compared with each other. These, and more comparisons, are done for the examples in the appendices. Forecasted data can be graphically depicted with the original data to see the performance of the forecasting. The data can be saved in a file for further investigation.

4.9 Comparing SAPS-ST and SAPS-II

SAPS-ST is more of a prototype than SAPS-II. It has never been designed for speed, but for a maximum of flexibility in changing code and trying out new things. Therefore, many internal checks are implemented in SAPS-ST. This has severe consequences for speed of execution as can be seen from appendix C in section C.5.

The advantages of SAPS-ST are as follows.

+ New searching paradigms are incorporated, and existing ones are functionally compatible.
+ The concept of a friendly user interface is taken further in SAPS-ST, where everything is window-based, and pull-down or pop-up menus respond ‘intelligibly’ depending on the progress made in the identification stage.
+ Everything is implemented in an object-oriented environment that is known to run under almost every known operating system [ObjectShare 1999]. Hence, porting the code to another platform is almost trivial (technically speaking, not financially).
+ Contrary to SAPS-II, no programming is required from the user.
+ More I/O variables and deeper masks can be tackled by the new sub-optimal search algorithm. It makes finding a sub-optimal mask much faster and it allows mask sizes that were not feasible before.

+ The implementation is object-oriented. The prototype is much more flexible, and even more modular than all previous SAPS implementations. New recoding and searching techniques can just be plugged in, and quality functions can be altered quickly and easily.

+ The mask evaluation is completely independent from the searching algorithms. The latter work with nodes, each node corresponding with a mask, its quality, and so on. The determination of quality happens in the node and is invisible to the searching routines. Hence, searching happens with abstract objects that respond by giving a quality value. A meta-node takes care of the generation of these objects [Van Welden 1999].

+ Crisp recoding is considered a special case of fuzzy recoding. Hence, recoding routines are only written once.

The disadvantages of SAPS-ST are

- Only three levels are implemented (no reconstruction analysis is included yet). This makes the prototype less useful at this stage.

- SAPS-ST is not as fast as SAPS-II. As a prototype, it is not intended to be. Moreover, many internal checks are implemented for debugging purposes in case one wants to change code. If it really has to become very fast, a conversion to C++ code may be advisable.

It is less tested and debugged than SAPS-II (because no team of users is working with it as is done with SAPS-II). Furthermore, as a prototype it is still prone to continuous changes. A version with a constant layout and user-interface is less likely.

4.10 Conclusion

From the arguments presented in this chapter, it is obvious that for systems that are more complex the optimal mask search algorithm may not be sufficiently fast, or may fail to find a suitable mask. A new sub-optimal mask algorithm, which is justified in this chapter, alleviates this problem to a great extent. It gives new impetus to further research where masks that are more complex may be considered. For example, derivatives may be included in the mask, more I/O variable systems can be tackled, and larger time constants can be captured in a deeper mask. A primary mask, usually a maximal allowable mask, puts an upper bound on the search space.

The sub-optimal search technique is implemented in a totally novel version of SAPS, called SAPS-ST. SAPS-ST is a partial, object-oriented version of SAPS that comprises of the sub-optimal mask search algorithm in juxtaposition with the exhaustive search algorithm[7]. SAPS-ST is also compatible with SAPS-II, but is user-friendlier and, as a prototype, particularly suited for extension with all kinds of new algorithms for recoding, mask quality determination, searching, and forecasting. Consequently, SAPS-ST is a prototype that provides fast ways to try out new things. It is by no means a competitor of SAPS-II.

[7] SAPS-ST also has some other search algorithms included that are not yet fully tested (e.g., simulated annealing and genetic algorithms)

References

Barreto J. [1994], Personal communication.

Cellier F.E. [1987], "*Qualitative Simulation of Technical Systems by Means of the General System Problem Solving Framework*", International Journal of General Systems, 13(4), p. 333-344, 1987.

Cellier F.E. [1991], Continuous System Modeling. Springer Verlag, New York, 1991.

MathWorks [1999], www.mathworks.com

Objectshare [1999], www.objectshare.com

Uyttenhove H.J. [1978], Computer-Aided Systems Modelling: An Assemblage of Methodological Tools for Systems Problem Solving. Ph.D. thesis, School of Advanced Technology, State University of New York at Binghamton, 1978.

Van Welden D., Vansteenkiste G.C. [1994], "*SAPS-ST: A Testbed For Incremental Research on GSPS*", Proceedings of the 1994 European Simulation Multiconference, Barcelona, Spain, June 1-3, p. 507-513, 1994.

Van Welden D., Vansteenkiste G.C. [1996], "*Sub-Optimal Mask Search in SAPS*", International Journal of General Systems, vol. 24, 1-2, p. 137-150, 1996.

Van Welden D. [1999], "*Compatibility between SAPS-ST and SAPS-II* (with User-Manual of SAPS-ST), Technical Report, Universiteit Gent, 1999.

Winston P.H. [1992], Artificial Intelligence, 3rd edition. Addison Wesley, 1992.

PART II

A Data Mining Approach to Identification of Dynamical Systems

Chapter 5

From Knowledge Discovery in Databases to Classification

5.1 Introduction

This chapter provides an overview of the emerging popular domain of Knowledge Discovery in Databases (KDD) and data mining. Although the latter is a process in KDD, it is often abusively interchanged with the former. This chapter will shed light on some terminology and furnish the necessary formalisations. KDD is described in a broad context (a 'bird's eye view'): its relation with data warehousing is highlighted and it is shown that the approach is goal driven. The life cycle (herein called virtual circle) of KDD, which is reviewed in chapter 7, is explained. The concept of supervised learning is borrowed from the field of machine learning. It covers classification and regression. Classification is based on the values in a classifying attribute while prediction predicts some unknown or missing attribute values based on other information. Both should preferable lead to the understanding of the variables and the interactions that drive the phenomenon. Anticipating chapter 6, this chapter will concentrate on classification, which basic purpose is to produce an as accurate as possible classifier or to provide insight and understanding into the predictive structure of the problem. Many principles underlying classification can be carried over to classification and regression trees, which form the subject of chapter 6. Some algorithmic classification methods, such as linear functions, logical descriptions, decision trees, neural networks, etc, are run through. This chapter can be compared in structure with chapter 1: it lays down the basic framework for the sequel.

5.2 Knowledge Discovery in Databases and Data Mining

Nowadays, more and more data is digitised and stored in databases thanks to the automation in generating and collecting of data (examples are tax returns, health care transactions, consumer behaviour, industrial transactions, etc.) and the maturity of database technology. The resulting data explosion cannot be processed by humans alone anymore (one is drowning in data) despite the fact that quite some informative patterns may be hidden in the data (but one is starving for knowledge). Consequently, tools and techniques have been designed that assist humans intelligently and automatically in analysing this abundant amount of raw data to obtain useful information via knowledge discovery in databases (KDD). Thus, KDD is a data exploration methodology that is defined to be the non-trivial extraction of implicit, previously unknown, potentially useful, 'relatively simple', and not predefined information[1] from large databases.

[1] The weight of the adjectives may vary according to the goal of the exploitation (see interestingness function)

KDD is employed in finance (fraud detection, stock market prediction, credit assessment), marketing (basket analysis, target marketing, sales prediction), quality control, medicine (effect of drugs, diagnosing, hospital cost analysis), astronomy (cataloguing), molecular biology (finding patterns in molecular structures), text mining, web mining, etc.

Economical criteria are 'les raison d'être' of KDD in business enterprises. Ultimately, it should give a good return on investment. Customer retention, fraud detection, risk analysis, management applications and market analysis applications are the driving forces behind KDD. *Practical criteria* express that it must have a potential for significant impact of an application. It should be advisable because no standard methods exist to solve the problem. In addition, there must be domain expertise available. Juridical criteria, such as privacy and tractability issues, play along. *Technical criteria* imply that there should be sufficient data available because many data mining algorithms require large amounts of data. From what has been described in the beginning of this section, it is clear that the presence of abundant data is becoming more a rule than an exception. Nonetheless, the variability in the data should be high enough and relevant attributes must be present.

In this thesis, emphasis will lie on the technical criteria.

KDD deals with 'real world' problems. Huge databases need scalable algorithms to be scanned for interesting patterns. With large data sets involving many variables, more structure can be discerned if a variety of different approaches are tried. However, the expansiveness by itself does not necessarily imply a richness of structure. Having many variables entails a high dimensionality, which may trigger the curse of dimensionality (sparse data). An effective medicine for this is the use of a-priori knowledge and data reduction techniques (see section 5.4.3). The huge number of data may also give rise to finding '*spurious patterns*': assessing statistical significance becomes very important [Salzberg 1997]. Additionally, non-homogeneity, missing and noisy data complicate knowledge extraction. Overfitting tends to destroy the parsimony and the generality of a pattern. This pattern should be time-invariant in the simple case. Time-variant patterns, present in non-stationary data such as ageing or obsolete data, complicate the search tremendously.

A definition of KDD is found in the glossary. Many adjectives are used to make it as precise as possible. Still, formalisation goes a step further by defining an interestingness function. The certainty (C), the novelty (N), the usefulness (U), the simplicity (S), the comprehensibility (V), the generality (G), and the redundancy (R) of a pattern E in a data set F are formalised and used as arguments in its definition, i.e.,

$$\text{interestingness } i = I(E,F,C,N,U,S,V,G,R)$$

Some measures are more objective, e.g., confidence, support, while others are more subjective, e.g., novelty, usefulness. The same is true for the interestingness function itself, which is goal dependent. A pattern is deemed knowledge if it is interesting enough, i.e.,

$$\text{if } I(E,F,C,N,U,S,V,G,R) > i_{\text{threshold}}$$

where the threshold is chosen by the user.

The interestingness function can be seen as a generalisation of the quality function in SAPS, which evaluates a time-invariant pattern, a mask, with regard to certain, sometimes quite

subjective, criteria. Defining an interestingness function may look easy, but how to fill it in is another matter.

5.3 KDD in a broader perspective: the virtuous cycle of data mining

KDD[2] is itself a sub-process of what is called 'The virtuous cycle of data mining'. It shows the complete enterprise approach to KDD. It goes on where KDD ends by transforming information into actionable processes. Berry [1997] writes that '*one must be able to respond to the patterns, to act on them, ultimately turning the data into information, the information into action, and the action into value*'. Put it otherwise, KDD is in fact only one major step in a process that applies knowledge gained from increased understanding of customers, markets, products, and competitors to internal (continuously evolving) processes.

Figure 5.1 gives an integrated and broad picture showing the three main societies involved in the virtuous cycle, i.e., the management society, the database society, and the ML-statistic (ML stands for Machine Learning) societies (ML and statistic is grouped here). It further shows that a data warehouse delivers data to three processes: report generation, OLAP (On Line Analytical Processing), and KDD. Low-level data processing occurs in a data warehouse, see section 5.4.1. It should precede report generation, OLAP and KDD.

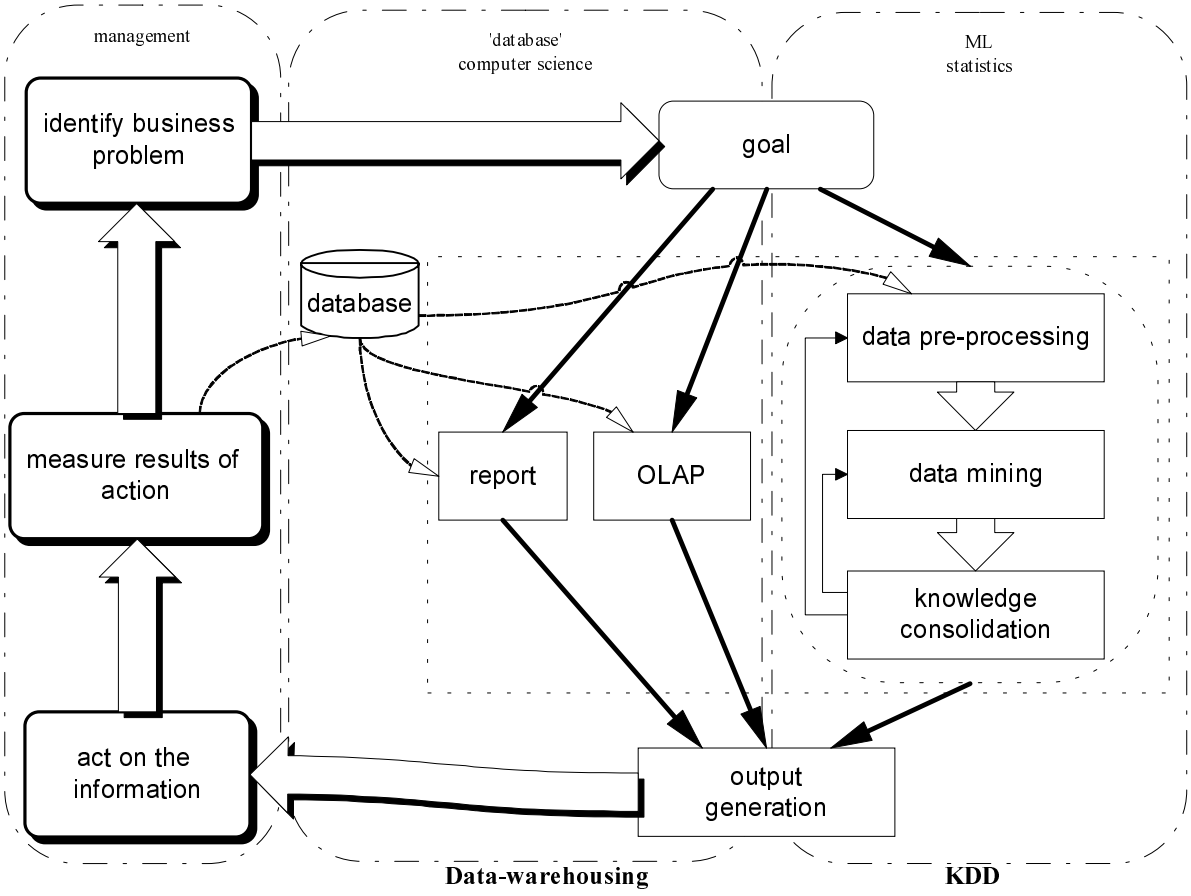


Figure 5.1 : Decision support and the role of data mining

[2] Berry[1997] calls KDD simply data mining

One could argue that data pre-processing and knowledge consolidation is equally present in the other two modules: report generation and OLAP. However, the explicitness and the importance of the pre-processing and knowledge consolidation are less elaborated than when doing KDD. E.g., usually, a report generation triggers a management action, but knowledge consolidation is less an issue for there is no new information generated. Pessimistically spoken; even OLAP is nothing more than a very powerful tool for reporting on data, in contrast to data mining that concentrates on finding new patterns in data, i.e. inducing knowledge. OLAP is a presentation tool that can permit manual knowledge discovery, but it relies entirely on human intelligence for the knowledge discovery piece, and this is not what is meant with KDD.

5.4 The basic steps in KDD

The basic steps in KDD are depicted in the KDD block in Figure 5.1. In utilising these steps, visualisation of the data is quite important (e.g., scatter plots for trend or outlier detection, etc.). At a meta-level, one should also consider the data warehouse, which has its own merits with regard to KDD. That is why it is included in this section as step 0. The remaining steps match largely with what is found in [Fayyad 1996].

KDD starts with the *goal definition*, which must be formalised and made executable so that it can be related to relevant data, which are hopefully present in the database. The *data preprocessing* step prepares and reshapes the data for subsequent processing. It involves data and attribute focusing, data cleaning, data projection, and data augmentation. The *data mining* step induces the model. It consists of a model specification, model fitting, model evaluation, and model refinement. Similarities with 'classical' modelling emerge. They will be discussed in chapter 7. *Consolidation of the newly found knowledge* and output generation consists of interpreting and documenting the found patterns, and if more model types were used, comparing them. Any conflicts (if any) that may arise with previous knowledge in the knowledge base must be resolved. A priori knowledge can be used in any step.

These steps can be illustrated with what an archaeologist does, see [Brachman 1996]. When an archaeologist wants to dig up interesting artefacts (goal definition), he looks at the (data) landscape to decide where to dig. This is based in part on what he sees and in part on his experience and background knowledge (focusing). Once at the site, he brushes away the dust (data cleaning and data projection), pieces fragments together that seem to fit (data mining), and decides what to do next in order to confirm an evolving hypothesis about the creator and the meaning of the artefacts (consolidate new found knowledge). Finally, the artefact is presented in a museum (output generation).

Remark the possible impact of different degrees of a priori knowledge in the process. More a priori knowledge helps in all steps to achieve a better (more valid) model (is the artefact a vase, an eating bowl, a religious symbol, etc.?). Remark that the model specification can be purely data driven too (just try to puzzle pieces together). Thus, data analysis and model development is highly intertwined.

Tools that only offer specific data-mining methods are plenty, but tools that do KDD (in its full meaning) are still rare. Three examples that approximate KDD are: SAS Enterprise Miner (from [SAS 1996], EXPLORA (from [Klösgen 1996]), and Darwin, (from Thinking Machines Corporation, [Think 1999]). For a current overview of the suites and tools, see [Siftware 1999].

5.4.1 Step 0: Collecting data in a data warehouse

A data warehouse collects data from different sources and integrates them in a meaningful way. Operational data can be stored in different databases in and different formats: relational, transactional, object-oriented, object-relational, active, spatial, time-series, text, multi-media, heterogeneous, etc. Hence, it is far from trivial to merge these data. When they come from different sources, an extra complication arises from the syntactical different representations for the same semantic data. For example, 'yes ' (with an extra space at the end) is not syntactical the same as ' yes' (even more syntactical different may be 0 for 'no' and 1 for 'yes'), although semantically they may very well be. Hence, data warehousing converts the data into a suitable form for report generation, for OLAP, and last but not least, for KDD.

5.4.2 Step 1: Goal definitions and problem types

A goal definition is the starting point for KDD. For a goal set forward, one has to formulate the problem and get an understanding of the domain in which it is situated. A priori knowledge, which may be present in a knowledge base, must be considered and used whenever possible. The goal should be formalised and made executable so that it can be related to relevant data. The goal for exploring data in a database can be different from situation to situation. At an abstract level, two high-level goal settings can be distinguished: hypothesis testing and knowledge discovery. Both are depicted in Table 5.1 (left versus right column).

Goal setting 1:	Goal setting 2:
Top-down	Bottom-up
Hypothesis testing	Knowledge discovery
Confirmative	Explorative

Table 5.1 : Synonyms for the two main approaches in KDD

Hypothesis testing is a top-down or confirmative approach. Once a hypothesis is stated, a list of data requirements has to be generated to test it. Unfortunately, even in the data warehouse the collected raw data is usually not stored as such to test hypotheses directly.

Knowledge discovery is a bottom-up approach. Undirected knowledge discovery is harder than directed knowledge discovery due to the extra level of possible class combinations. Hence, undirected knowledge discovery tries to recognise relationships while directed knowledge discovery tries to explain those relationships. *Undirected knowledge discovery is often a prelude to further investigation with directed knowledge discovery.*

One may consider some typical goal types with their corresponding problem types.

A first goal type is to gain insight or to have a good model. This can be achieved through confirmation of a hypothesis (is insight true?) or via data exploration. It corresponds with the Summary and Description problem type. It may also lead to refinement of an existing model [Simoudis 1996]. The Nuggets problem type of Klösgen [1996] may be linked with this goal type.

A second goal type aims at a correct *classification* (classify on attributes) and *prediction* (fill in unknown values). This typical needs a data exploration approach, which corresponds with the 'Classification and Prediction' problem type (when time is less the issue). This case may also apply when the classification results are further processed by another automated method

(e.g. another computer program). A good example is where a qualitative model (as in SAPS) is built, and when this model is used to forecast: the syntactic complexity doesn't matter much, nor does the comprehensibility of the pattern.

A third goal type emphasises a speedy classification and prediction. This mostly needs a data exploration approach, but sometimes a hypothesis for classification may help. The corresponding problem type is the 'Quick decision' problem type (e.g. should I give a credit or not?). In this case, accuracy may be less important than speed and comprehensibility, e.g., neural networks.

Consequently, a goal setting has a significant impact on the a priori selection of data mining techniques.

This thesis focuses on the directed knowledge discovery approach.

5.4.3 Step 2: Data pre-processing

A hypothesis needs a list of data requirements to be generated for its testing. Unfortunately, collected data is usually not directly suitable for this purpose. Often, the data has to be prepared and perhaps transformed to be of good use. In the knowledge discovery approach, one may have not the interesting (and often unknown) data available in the database. Sometimes, the sheer number of data levels can make things explode combinatorial, creating the necessity for lumping data. Furthermore, one has to be aware of a possible data-suitability problem. The data pre-processing step tries to answer these problems via the application of two main steps. These are data cleaning and data reduction. For the latter, one distinguishes a vertical and horizontal reduction. This is because, usually, data is represented in a matrix: the columns represent the variables and the rows represent the data records. Consequently, data reduction can be achieved in two dimensions: horizontal reduction or dimensionality reduction (or feature selection), and vertical reduction or numerosity reduction.

Attribute focusing involves a horizontal data reduction in the parameter space. It creates a target data set by focusing on a subset of variables. It can be compared with an action in an experimental frame (chapter 1). Attribute focusing may also imply the right selection of predefined classes in supervised learning, see 5.5.2. If one is interested in only one class from many possible ones, then one may re-code the attribute values such that only two classes remain. This is a typical sub-problem type for the pure classification/prediction approach [Klösgen 1996].

Data focusing or numerosity reduction gives a vertical data reduction in the observation space. One technique to achieve this is discretisation. It can be applied for very large databases (terabytes). There is static and dynamic discretisation[3]. In data pre-processing, only static discretisation is considered. Notice the resemblance with the recoding techniques used in SAPS, an issue that is dealt with in chapter 8. Discretisation can also contribute to attribute focusing via data focusing (if discretisation reduces the number of levels to 1).

Data cleaning is a double-edged sword: it is usually required to 'clean' poor data, but it may throw away a crucial indicator by considering it as an anomaly. For example, when a customer is deceased, or when fraud is involved. Some data mining tools can work with missing values, others do not. Records with missing data fields may be thrown away, but useful information may go away too. Another solution is to put an estimate in the empty field. In more

[3] In fact, it is better to speak of quantisation, but in the KDD domain, one is accustomed to 'discretisation'.

seldom cases, the missing field may also contain an important indication of something. Handling missing values can be combined with data cleaning (especially if the data mining method does not support missing values). An information theoretic approach to data cleaning is presented in [Guyon 1996].

Data projection accomplishes a dimensionality reduction. However, there is often also the necessity for data augmentation. The resulting use of more variables is not considered contraproductive. Berry [1997] states: *"Instead of carefully choosing the few independent variables that you expect to be important, the data mining approach calls for throwing them all into the hopper and letting the data mining tool itself determine what is important"*.

5.4.4 Step 3: Data mining

The data mining step comprises model specification, model fitting, model evaluation, and model refinement. This section is not elaborated much because data mining will be discussed in more detail from section 5.5 on.

Model specification involves deciding whether to use classification, regression, neural networks, or other techniques. Hence, it determines the model type on which the goal type has a significant impact. *Model fitting* determines the values of some specific model parameters based on a chosen training set. Model specification and fitting deals with the model representation. The latter supplies a language for describing discoverable patterns. For example, a decision tree representation using univariate (single-field) node-splits cannot discover $y = x$ no matter how much data is given. Thus, representational assumptions inherent to a particular method need to be understood. This is much related to model search, which is used in a 'restricted' family of considered models (e.g., classification trees). Implementations of model search methods tend to use heuristic search techniques. Then, a parameter search is used to optimise the parameters with regard to a model evaluation criterion. For models that are more complex, a closed form is not available; greedy methods are used; e.g., see the approach in [Van Welden 1996; Van Welden 1998].

Model evaluation estimates how well a particular pattern meets the criteria of the KDD process. Evaluation on predictive accuracy is based on cross-validation or on a simple training-test set paradigm. Both concepts are explained in section 5.9.6. Evaluation of descriptive quality involves predictive accuracy, novelty, and comprehensibility of the fitted model via the interestingness function.

Model refinement, depending on the evaluation results, iteratively refines the initial model or tries other model paradigms.

5.4.5 Step 4: Data post-processing or knowledge consolidation

Many data mining methods exist. Examples are classification, regression, clustering, probability density estimation, summarisation, dependency modelling (this is what SAPS does), change and deviation detection, basket analysis (the list is not exhaustive).

Each candidate data mining method is post-processed to achieve the goal as optimal as possible. Discovered knowledge can be consolidated by selecting the best method from the different paradigms, by summarising and documenting the selected method(s) and by resolving possible conflicts with previous knowledge.

Before interpreting the found patterns one should determine whether they represent interesting knowledge and if the found knowledge is consistent with the goal setting. The latter is not as trivial as it seems. As an example, suppose that the goal is to find a comprehensible knowledge structure. Rules and decision trees fulfil this requirement better than neural networks.

Often, this can be dealt with in the initial steps. Nevertheless, even a decision tree (e.g., which was chosen to be the most comprehensible knowledge form) may be correct and computationally efficient, but not understandable by humans. Many authors [Gaines 1996] state that such a structure does not qualify as knowledge. This is a typical problem with inductive models: they do not give insight into the basis of decision making. Smaller coherent and familiar structures that give a better mapping with human cognitive models should be put forward (a ML viewpoint).

It is of paramount importance to realise that data mining results change over time (model change). Especially in the business domain, models expire and become less useful as time goes by. This is related to the fact that data age quickly. Markets and customers change quickly as well. Even scientific discovery is an ongoing process.

5.5 Learning Systems and KDD

The knowledge discovery approach in KDD, which tries to recognise patterns in data, is a form of inductive learning. Therefore, one can speak of learning systems. The latter term originates from the domain of machine learning (ML). However, KDD and ML differ in some aspects:

- KDD is about finding understandable knowledge, while ML is more concerned with improving performance of an agent (e.g., training a neural network to balance a pendulum is part of ML, but not of KDD)
- The efficiency of the algorithm and scalability is more important in KDD, because it is concerned with very large, real world databases, while ML typically looks at smaller data sets. Hence, associated real world problems such as missing values, large databases, noise, etc., are more studied in KDD.
- On-line learning is more familiar in ML than in the statistics domain (a sub-domain of KDD). The latter works often with off-line learning.

Clearly defined relationships are hard to specify, because KDD uses methods from ML. Hence, for all practical purposes, one could argue that ML is a sub-domain of KDD (in the large).

In learning systems, one starts from the point that expertise or accumulated experience is often in the form of records of (hopefully) correctly solved cases that form the sole source of knowledge. A drawback, however, is that heterogeneous knowledge or experience from an expert, which may help in solving a problem, is often not accounted for in this paradigm. This can be contrasted, on the other hand, with an expert system, which solves problems using a computer model of expert human reasoning. It this approach, it is easy to codify the ‘rules of thumb’ of an expert. A comparison between learning and expert systems is given in Table 5.2.

Learning system	Expert System
data-driven	knowledge-based
any type of algorithm	usually based on models of human expert reasoning
can exceed performance of experts	limited ability to exceed performance of experts
induce knowledge and may discover new relationships among concepts	deduce knowledge
difficult to insert heterogeneous knowledge from expert	codes expert knowledge, which may be very heterogeneous
constructed by data-mining engineers	constructed by knowledge engineers

Table 5.2 : Learning systems versus expert systems

Both approaches are complementary and should be combined to have a more powerful approach. One possible approach is described by Heckerman [1996], who examines a knowledge representation form, called Bayesian network, which integrates the best of both worlds. The method starts with an encoded representation of an expert's knowledge. The database of cases is then used to update this knowledge (inductive correction). Hence, the original expert knowledge is refined and new relationships may be discovered. The approach appears to be robust to errors in the initial a priori scheme. The result is an inductive, easy interpretable corrected a priori expert knowledge representation.

ML people soon reckoned the problem of induction. It suggests (Karl Popper) that scientific hypotheses can only be falsified. (Un)fortunately, when falsified, a new theory has to be built and usually this is done by aid of induction. The fact that one is never certain in an infinite input space that the correct hypothesis is reached has a great impact on computational learning.

5.5.1 Computational learning

The primary concern of computational learning is the development of learning algorithms. No programming is required during the learning phase. Computational learning roughly divides into three different areas: ML, connectionism (parallel-distributed processing or neural networks) and statistical learning techniques.
The learning process itself consists of choosing or adapting parameters within a chosen model type that work best on the samples at hand and others like them. Most current theory of machine learning rests on the crucial assumption that the distribution of the training examples is identical to the distribution of the test examples. In choosing the samples, one further distinguishes:

- Direct training examples versus indirect training examples. This can be compared with directed and neutral systems (chapter 1).
- The degree to which the learner controls the sequence of training examples; the sequence of examples may be under control of the learner or just given to him. This is comparable with actively or passively obtained data (chapter 1).

When the learning task is prediction, the goal of a learning system is prediction *on new cases*, not discrimination between the existing sample cases. Discrimination (a type of classification) is an easier process than prediction (overfitting is less a problem in that case). The examples

in the appendices all look at the prediction performance on a test set, which is representative for the new cases. Features that are no more predictive than chance can be considered noise. A problem is that such features, which appear noisy on their own, may prove to be highly predictive when combined with other features. For example, the entropy may decrease very suddenly by removing an attribute; a goodness-of-fit may also change drastically by adding or removing a predictor[4]; or a decision tree may become much simpler by removing a feature. Hence, performance of a learning system can be drastically affected by the choice and specification of the features (source system considerations in GSPS).

5.5.2 Supervised learning, unsupervised learning and target mapping

Unsupervised learning has as its objective to establish the existence of classes or clusters in a given data set. Supervised learning tries to induce a rule whereby one can classify a new observation into one of a set of existing predefined classes. In supervised learning, computational learning boils down to the construction of a target mapping that enables correct outputs to be returned for inputs that do not appear in the training set [Thornton 1992]. Each example is a training pair and shows a single association in the mapping and is divided up into two parts:

1. the first part is a list of symptoms or attribute values for a certain case,
2. the second part contains the desired output or target output.

The set of data records, which provides the necessary examples, is supplied to a suitable learning algorithm. The examples should enable the derivation of a general rule that allows correct diagnoses to be produced for any particular case, not just the ones explicitly shown. That is to say, they should enable to make an inductive leap from the specific to the general case. The target mapping guides in the search for the general rule just described. However, most target mappings are non-operationally defined. They have to be put in an operational form (evaluation function) that is easier to learn. Hence, the learning task is reduced to the problem of discovering an operational description of the ideal target mapping, usually in a functional representation (function approximation), i.e., a kind of optimisation. This can be compared with SAPS, where the general rule is the (sub)optimal mask and the optimisation process is the mask search.

5.5.3 Inductive bias

The power of a model depends on its specification. For example, a conjunctive expression is represented in the instance space (feature space) by a simple rectangle, while a disjunctive expression is represented in the instance space by a union of two simple rectangles. If only conjunctions are used, patterns that efficiently can be represented by e.g. disjunctions cannot always be represented. It is said that the learner is biased to consider only conjunctive hypotheses. The obvious solution is to enrich the hypothesis space to include also disjunctions and negations.

This leads to a very important fundamental property of inductive inference [Mitchell 1997]: *A learner that makes no a priori assumption regarding the identity of the target concept has no rational basis for classifying unseen instances.* This a priori assumption is what Mitchell calls the inductive bias. Its role is depicted in Figure 5.2. It is the policy by which a learner generalises beyond the training data.

[4] Terminology changes according to the domain (statistics, machine learning, system dynamics).

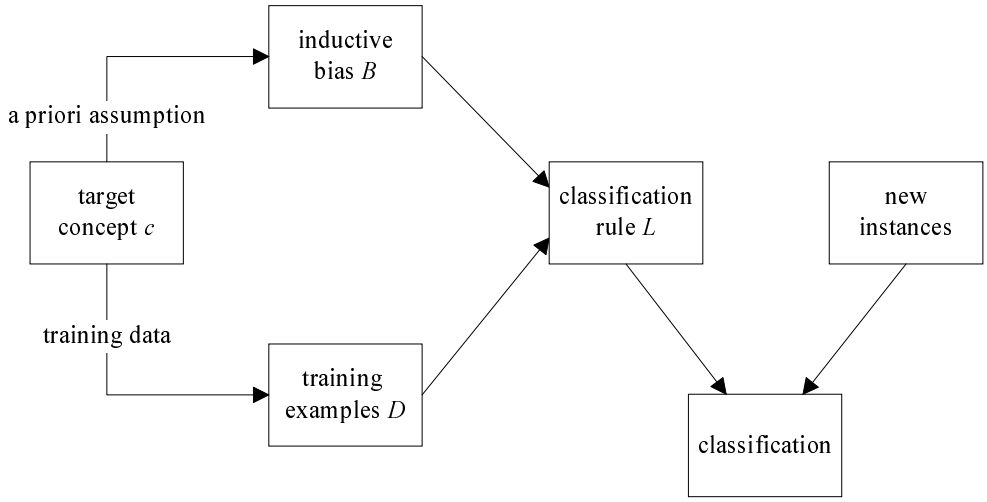


Figure 5.2 : Inductive bias

Hence, inductive bias characterises different approaches to inductive inference. Example:

- for a candidate-elimination algorithm, the inductive bias is that the concept *c* to be learned is included in the hypothesis space *H*.
- for tree classifiers (the subject of chapter 6), the inductive bias is a preference for small trees over large trees.

5.5.4 Classifiers

Classifiers are kinds of learning systems. There are two very different ways a classifier can be constructed. On the one hand, the classifier might be obtained by interviewing the expert(s). Alternatively, numerous recorded classifications might be examined and generalised by discovering and analysing patterns in them, from which a classifier may be constructed inductively. The former approach is knowledge-based while the latter is data-driven (see Table 5.1). In classification theory, it is assumed that a classification procedure is needed that will be applied to a continuing sequence of cases, in which each new case must be assigned to one class on the basis of observed attributes or features. Usually, it cannot be answered in advance which classifier performs best for a given real problem (this problem is of course equally present in KDD). Especially when the data are not well known, one has to try out a variety of methods on the available data and empirically compare the results. Furthermore, different problems have different characteristics and may require different classifiers. In searching for the best classifier, one has to express the goal type clearly.

5.6 Classification principles

Usually, sample data consists of a finite set of samples of solved cases. They imply a pattern of observations and the corresponding correct classification. The simplest view of the task of classification is that a specific pattern of observations is associated with a specific class. This is the pattern recognition perspective. The key requirements for a classifier that are assumed in this thesis are

- *attribute-value description*: all information about one case must be expressible in terms of a (fixed) collection of properties or attributes (attribute value learning).

- *crisp discrete classes:* the categories to which cases are to be assigned is established beforehand as in supervised learning. The classes must be sharply delineated (crisp); each case belongs or does not belong to one class (mutually exclusive classes). Fuzzy memberships are not considered. A classification rule should mimic this (unknown) function as much as possible.
- *non-parsimony:* there must be far more cases than classes. Specifically, there must be more than one case per class. Otherwise, classification does not make sense.

Other requirements are found in KDD: e.g., sufficient data, relevant attributes, etc.

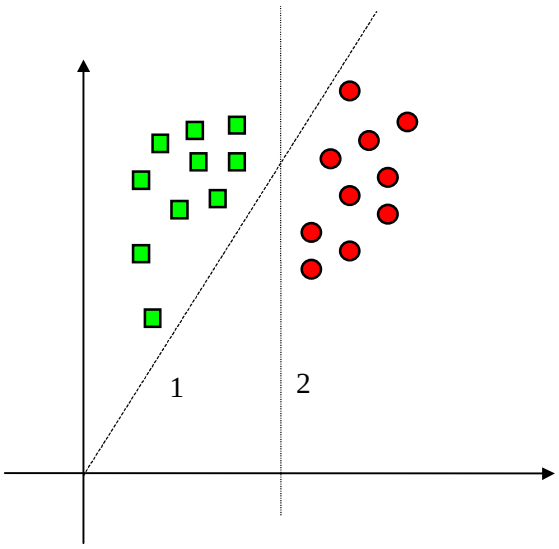


Figure 5.3 : Two distinct separators

In Figure 5.3 two distinct separations are drawn, one which could be the result of a linear discriminant analysis (1), and one which could stem from a split in a decision tree (2). Which one is the best for forecasting when new data will become available? From Figure 5.3 it can be noticed that a different boundary and consequently, a different interpretation is possible. The question is then whether a simple explanation for the split is more important than good forecasting. This, of course, is linked with the form of the interestingness function.

In practice, some (partial) requirements are often taken as a criterion for the quality of a classifier. Some theoretical requirements (accuracy, resemblance, etc) are formalised rigorously in [Van Welden 1998], but will not be discussed here. This section treats practical aspects that will be equally applicable when comparing GST and KDD.

A first important requirement of a classifier is its *comprehensibility*. Comprehensibility has to do with the colour of a classifier model. *Accuracy*, often the most important requirement, will be formalised later in section 5.9.4. It is important to consider accuracy classifications not only on a training set, but also on a test set. Only the latter is unbiased and truly a good measure. In some applications, *speed* is more important than accuracy if the difference in speed does not correspond with a too drastic difference in accuracy. An example is the automatic reading of postal codes in a large post centre. *Adaptability or time to learn:* in a rapidly changing environment, a classification rule must be constructed quickly with only a relatively small number of observations (model switching in control processes). In this context, one can distinguish between eager evaluation and lazy evaluation. These concepts come back in chapter 8. *Cost of classification:* The cost of test should be taken into account when con-

structing classifiers. Sometimes, not all tests are necessary. Especially when tests are relatively expensive, it may be advisable to lower (total) costs by alternating test and classification decisions as can be done in tree-classifiers. The *robustness* of a classifier plays a central role when noisy data is present. Finally, *scalability* of a classifier is an essential but relatively new concept in the context of data mining. A classifier can perform well on small data sets, but may fall short when trying to classify the typical huge data sets in KDD.

5.7 Classification approaches

If unlimited data were available, each pattern could be stored in a table for look-up purposes. One would simply look up in the table the corresponding class that had previously been associated with it (efficient look-up techniques such as hashing could be used to speed up the process). However, even the most systematic and long-term record keeping is unlikely to cover all the possible combinations of values that can arise in nature, see section 5.5. Thus, the learning task becomes one of finding some solution that identifies essential patterns in the samples that are not overly specific to the sample data (see section 5.9.2). The machine learning approach to this task is somewhat different from the statistical approach.

5.7.1 The machine learning approach versus the statistical approach to classification

The machine learning approach has some emphasis shifts compared to the statistical approach. They relate to the mental fit of the found models, the handling of noise, and the type of variables they focus most on.

Machine learning was first applied in agriculture to develop rules for diagnosing soybean diseases [Michalski and Chilausky 1980]. Rules derived by experts and those generated by a machine-learning algorithm were put side by side. When tested in an expert system, the machine-generated rules outperformed distinctly those generated by experts. It did not only give a good fit to the data, but also a good mental fit [Michie et al. 1994]. This 'knowledge' orientation is so important that data-derived classifiers, however accurate, are not ordinarily acceptable in the absence of mental fit, [Feng 1994].

The statistician viewpoint to classification is expressed in [Breiman 1984]: "*Depending on the problem, the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem*".

Hence, machine learning drives the mental fit concept much further than the statistical approach does. That is why ML gained earlier acceptance and why decision trees became more popular in that field despite the emphasis on noiseless cases. However, the statistical approach is more realistic in considering the presence of noise. Contrary to the noiseless situation, measurement points do belong to different classes and no unique membership is guaranteed.

This thesis adheres to the statistical approach.

Accordingly, a way to look at classification, is by starting with the assumption that the design set is a random sample from the overall population. An overlap between the probability distributions of the different classes is an inherent aspect of the statistical approach. A decision surface or line will be based on a threshold. The best one can do is to approximate the Bayes' classification rule.

One thus considers the density function $p(j|\vec{X}=\vec{x})$, which represents the discrete class distribution conditional on a measurement vector $\vec{x}$. In the case of the ideal situation the density function would be a Dirac function on the classes, i.e., the probability is 1 when $\vec{x}$ belongs to class *j*, otherwise it is 0. The difference between the two approaches is depicted in Figure 5.4.

What are searched for are estimators of $p(j|\vec{X}=\vec{x})$ that are as close to the true values as possible. However, it can not be done by looking at some measure of difference between estimate and true value, because the latter is not known. Information about the true values can only be obtained via the design samples. Consequently, many classification methods are based on estimation of $p(j|\vec{X}=\vec{x})$ or $f(\vec{x}|Y=j)$ (via the Bayes rule). The latter density is often denoted by $f_j(\vec{x})$.

- Parametric methods assume certain distributional forms (e.g. discriminant analysis in general assumes multivariate normal distributions).
- Non-parametric methods do not make the latter assumption; they are based on vicinity considerations (e.g. nearest neighbours).
- Intermediary methods are methods that have a very large number of parameters. An example is neural networks, [Hand 1997].

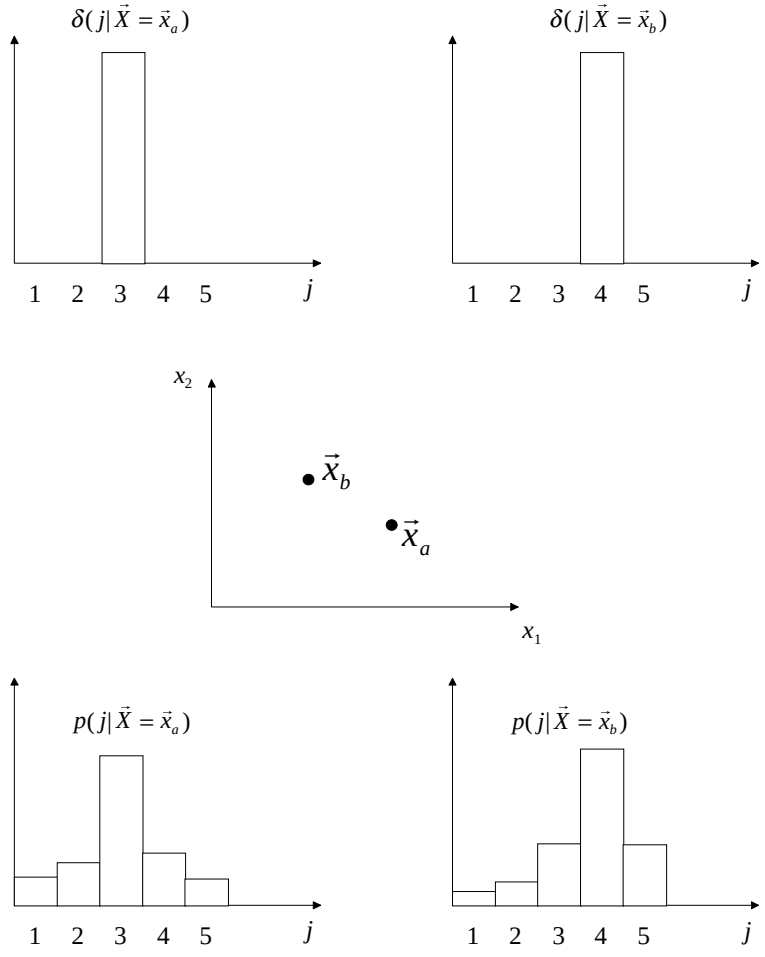


Figure 5.4 : Machine Learning (top) versus Statistical Approach (bottom)

A smaller emphasis shift is that statisticians often think in terms of continuous variables. In contrast, the ML community are more used to discrete (and binary) variables. Fortunately, this has not restricted too much the applicability because most problems are categorical or can be categorised. An illustration of this is found in chapter 6, where the most prominent decision tree algorithm from the ML world works with discrete classes (e.g., classification trees), while in the statistics world an extension to regression trees is made.

5.7.2 Taxonomy of classification methods

The methodological differences between machine learning and statistics are found back in a taxonomy of classification methods. Their strands of research that can be identified for classification can be found back in Figure 5.5.

- Statistical pattern recognition :
  - Bayes rule
  - Discriminant analysis
  - Nearest neighbour method
  - Density estimates
  - Tree classifiers: CHAID, CART
- neural networks (connectionist approach)
- machine learning methods
  - Concept learning and rules
  - Tree classifiers: ID3, C4.5, C5.0
  - Case based reasoning
  - Genetic algorithms

Figure 5.5 : Taxonomy of classification methods

The taxonomy can be elaborated upon by distinguishing between

- linear, quadratic, and logistic discriminant analysis,
- the kinds of neural networks : backpropagation network, Kohonen network, …
- genetic algorithms and evolutionary algorithms,
- variants on existing algorithms such as CART and C4.5,
- etc.

Other taxonomies can be devised. For example, Quinlan [1993] takes neural nets and genetic algorithms as a part of machine learning. Yet other taxonomies are found in [Henery 1994] and [Info. Disc. 1998].

5.8 Popular classifiers

In comparing classifiers, a clear distinction is made between the performance of a classifier rule, induced on a particular design set, and the performance of the classifier method itself (compare with the difference between estimate and estimator accuracy). The former has to do with how well this particular rule performs given a certain classifier model, while the latter has to do with what type of rule (model) is constructed. Thus, the former is conditional on the latter (model). The methods for solving the classification problem are domain independent. They assume little or no knowledge of the semantics of the application area. All the observations are merely treated as symbols that are manipulated by a computer.

5.8.1 Concept learning

Concept learning encompasses acquiring the definition of a general category given a sample of positive and negative training examples of the category. It infers a Boolean-valued function from training examples of its input and output.

The target concept c to be learnt is a Boolean function $c: X \rightarrow \{0,1\}$, where X is the set of items over which the concept is defined. The training examples D consist of an instance x from X, along with its target concept value $c(x)$, i.e., $\langle x, c(x) \rangle$. Positive examples are instances for which $c(x) = 1$, negative examples are instances for which $c(x) = 0$. Given D and examples of c, estimate c or find a hypothesis h in a hypothesis space H such that $(\forall x \in X)(h(x) = c(x))$. The only information available about c is its value over the training examples. Therefore, inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data. Concept learning can now be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation, where the goal is to find the hypothesis that best fits the training examples.

Patterns are used to characterise a hypothesis. Taking as a convention that the hash character '#' is a wild-card that stands for any valid symbol from a certain alphabet ({0,1} in the binary case), the hypothesis [1 #] matches the mapping in Table 5.3 in which the output for [1 #] is considered 1 or true.

	input			output		
<	[0	0	]	[0	]	>
<	[0	1	]	[0	]	>
<	[1	1	]	[1	]	>
<	[1	0	]	[1	]	>

Table 5.3 : The mapping corresponding with [1 #]

By selecting a hypothesis representation, the space of all hypotheses is defined.

5.8.2 Linear classifiers

A linear classifier assumes that a class can be expressed as a linear combination of the attribute values. It tries to find a particular linear combination that gives the best fit over the training cases. Linear discriminant analysis uses weighted contributions of attributes that are combined arithmetically and compared to a threshold. In linear discriminant analysis, one assumes

that all $f_j(\vec{x})$ are multivariate normal densities with common covariance matrix and different means vectors. Other methods, which require fewer assumptions about the form of the underlying distribution, are the nearest neighbour and density estimate methods.

5.8.3 Tree classifiers

A tree classifier is a structure that is either

- a leaf indicating a class (or distribution of classes), or
- a decision node that specifies some test to be carried out on a single attribute value, with one subtree for each possible outcome of the test.

A tree classifier is used to classify by starting at the root, moving down and performing a test (decision) at each node until a leaf node is reached. The unique path from the root to each leaf can be considered as a rule classifying the records. Many different leaves may produce the same classification, but the path from the root is different, thus the reason for that classification is different too.

Tree classifier applications are not limited to the development of accurate predictors: they should also be intelligible to humans. In the latter case, unwieldy trees are out of the question. A possible solution is one in which a large tree is broken down in a hierarchy of smaller trees that are individually and collectively better to understand. Chapter 7 elaborates on tree classifiers, so a detailed description is postponed until chapter 7.

5.8.4 Relational learning models or rules

Relational learning models are based on first-order logic. Logic expressions can work well with both numeric and non-numeric data. Patterns expressed as logical languages are readable and understandable. They are good for representing crisp boxes.

There are *conditional rules*, where statements are of the form

if A then B

There are also *association rules*, where statements are of the form

when A also B

Association rules express statements like '*x* % of people that do this also do that'. Agrawal [1996] explains two new algorithms that outperform older algorithms by an order of magnitude for large problems and scale up linearly with the number of transactions (scalability issue). The prototypical application is analysis of sales data. Decision trees cannot express associations.

Attribute logic compares attributes. This kind of logic is usually embedded in conditional or association logic. Attribute expressions are of the form

A = B

or, A < B, or, A > B, or, A ≤ B, or, A ≥ B. Attribute logic cannot be expressed in decision trees, because there is no explicit naming of values.

Of course, combinations of all above type of rules are possible, like

(if A then (when B also C)) with confidence D %

5.8.5 Non-linear classification methods

Non-linear classification (and regression) fit linear and non-linear combinations of basis functions to combinations of input variables. Examples are neural networks, adaptive spline methods, non-linear discriminant analysis, (and non-linear regression), etc.

Neural networks are based on a model of the human brain. They are like black box systems. The patterns they learn are stored in the weights of the interconnection between the neurones. Neural networks of an appropriate size can approximate any smooth function.

Genetic algorithms are based on evolutionary models in which 'survival of the fittest' is in order.

Both methods are very powerful in representational power, but difficult to interpret.

5.8.6 Example-based methods

Example-based or instance-based classifiers rely on similarity properties. A case can be classified as belonging to a class that has cases that are most similar to it. Ideally, prototypical cases should be retained that summarise all the important information. Generalisation beyond the training examples is postponed until a new instance must be classified. The similarity of a new instance with instances in the training set is usually based on a metric on important attributes. So, there is no natural or simple way to handle categorical variables and missing data. Example-based methods use representative examples to approximate a model. Prediction on new examples is based on similarities with existing examples whose prediction is known. Nearest-neighbour classification, kernel density estimation, locally weighted regression algorithms and case-based reasoning belong to the example-based methods. Finding a well-defined distance metric is not always an easy chore. Example-based methods are very powerful in their representation but difficult to interpret since the model is implicit in the data. Example-based methods are sometimes referred to as 'lazy' learning methods. Hence, they do not construct a general, explicit, description of the target function from a training set. Instead, they estimate it locally and differently for each new instance.

Of particular interest in this thesis is the k^{th} nearest neighbour rule (kNN). For a chosen metric and a fixed positive integer k, the k nearest neighbours are searched and the data point is classified in class j if more of these neighbours are in class j than in any other class. This data retention approach has no natural or simple way to handle categorical variables and missing data. Furthermore, the approach is computationally expensive and gives very little usable model information regarding the structure of the data. The kNN method is applied in SAPS. It will be discussed in chapter 8.

5.8.7 Probabilistic graphical dependency models

Probabilistic dependencies underlying a particular model can be put in graphical models that specify which variables are dependent on which other ones. In the AI (Artificial Intelligence) community, this approach was knowledge driven: the conditional probabilities attached to the links of the graph were elicited from experts. Now these probabilities can be learnt from the data itself. The graphical form leans itself for easy interpretation.

A Bayesian network is an example of a probabilistic graphical dependency model. It encodes (a priori) expert knowledge and refines this representation from induced knowledge (from the data). In contrast with e.g. neural networks, Bayesian networks have a great advantage concerning its interpretability and its encoding ease of a priori expert knowledge [Heckerman 1996]. The use of causal relationships is what makes the Bayesian network so easy to com-

prehend. A problem with Bayesian networks is that it depends on variable order. The order is chosen by an expert; it shows the role of the expert knowledge in exploiting causal relationships among variables. If, however, the expert knowledge is unreliable or incomplete, the data can be used to update the structure or learn probabilities.

5.8.8 Temporal pattern detection

Much of the data stored in databases is temporal, i.e.; the variables evolve in time. A descriptive approach consists of analysing patterns, while a predictive approach is about forecasting of events. Of course, the former can sometimes be used for the latter. In general, one distinguishes three kinds of approaches to time series, which shows how KDD looks at temporal (dynamic) data.

In *trend and deviation analysis*, one seeks to characterise an evolution in trend, sequential patterns, similar sequences, and deviation data, e.g., stock analysis

In *similarity-based pattern-directed analysis*, one attempts to find and characterise user-specified (sub) patterns in large databases. Typical applications are found in the financial market, in basket data analysis, in scientific databases, in medical diagnoses, etc. Comparison is done of characteristic features such as the Fourier transform, see [Faloutsos et al. 1994] or via time warping, [Berndt 1996]. Even a special pattern language to encode queries about 'shapes' is described in [Agrawal et al. 1995].

In *cyclicity/periodicity analysis*, one tries to find segment-wise or total cycles or periodic behaviours in time-related data

5.9 Validation of classifiers via train and test samples

When a classifier rule is invoked, it will classify a proportion of the future (new) samples in the right class, but it may also misclassify another proportion. If the classes are not separable and if the precision is not perfect there are bound to be an inherent proportion of misclassified cases. In addition to just counting the number of misclassifications, one can weight them in importance.

The concept of train and test samples allows a distinction between apparent and true estimates of error rate of a classifier. The next subsections formalise and elaborate on this distinction. First, some basic definitions are given from the viewpoint of statistics (in agreement with the statement in section 5.7.1).

5.9.1 Definitions and formalisations

The aim of a classifier is to decide to which class a case, characterised by a measurement vector, should be assigned. The measurement (feature) space X is defined as containing all possible measurements vectors $\vec{x} = (x_1, \cdots, x_n)$, where x_i is a measured variable (feature, attribute).

Suppose that the cases fall into J classes and that these are numbered from 1 to J. Let C be this set of classes, i.e., $C = \{1,2,\cdots,J\}$. A classifier or classification rule is a function $d(\vec{x})$ defined on X so that for every $\vec{x}$, $d(\vec{x})$ is equal to one of the number 1, 2, …, J.

$$\begin{aligned} &d\colon X \to C \\ &\vec{x} \to j = d(\vec{x}) \qquad\qquad j \in C \end{aligned} \tag{5.1}$$

Hence, it is a rule that assigns a class membership in C to every measurement vector in X. The rule can also be defined for a random variable (vector). The actual class of a measurement

point $\vec{x}$ is defined by $C(\vec{x})$. It is either that $d(\vec{x}) = C(\vec{x})$ (a right classification), or $d(\vec{x}) \neq C(\vec{x})$ (a misclassification). In systematic classifier construction, past experience is summarised by a learning sample[5] L consisting of data already observed and classified. Thus,

$$L = \{(\vec{x}_1, C(\vec{x}_1)); (\vec{x}_2, C(\vec{x}_2)); \cdots; (\vec{x}_n, C(\vec{x}_n)); \cdots; (\vec{x}_N, C(\vec{x}_N))\}$$

The learning sample consists of cases that are independently drawn from the relevant population. It is used to construct the classifier rule d in equation (5.1). Another sample is employed to validate the classifier model.

Alternatively, a classifier can be defined as a partition of X into J subsets, say $A_1, \cdots, A_J$, such that for every $\vec{x} \in A_j$ the predicted class is j, i.e.,

$$A_j = \{\vec{x} | d(\vec{x}) = j\} \qquad (5.2)$$

For a two-dimensional measurement space X, a partition may look like in Figure 5.6.

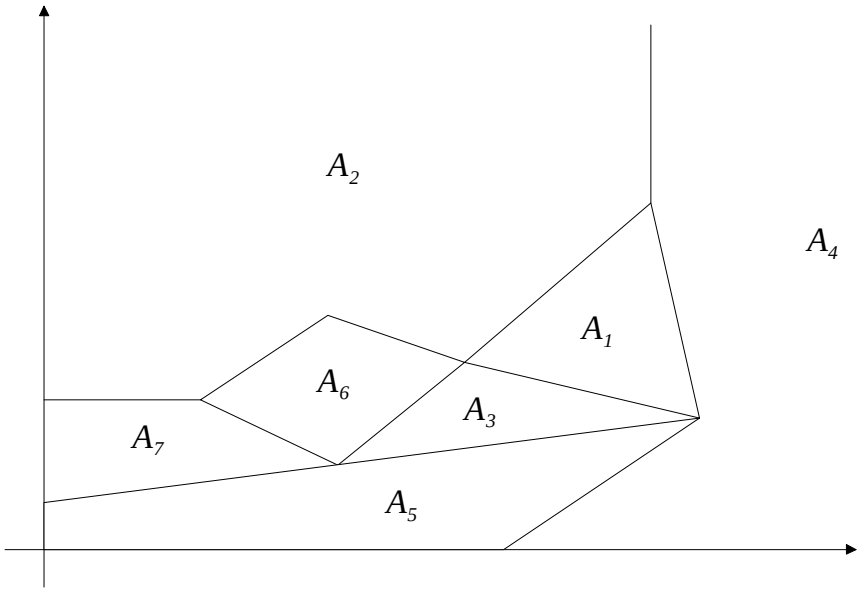


Figure 5.6 : A two-dimensional partition example

A case drawn at random from a relevant population has a probability of $P(A, j)$ that its measurement vector $\vec{x}$ is in A and its class is in j :

$$P(A, j) \equiv P(\vec{X} \in A, Y = j) \qquad A \subseteq X, j \in C$$

A probability model can then be defined on the space $X \times C$ consisting of all couples $(\vec{X}, j)$, i.e.,

$$\begin{aligned} P&: X \times C \to [0,1] \\ &(\vec{X}, j) \to P(A, j) \qquad A \subseteq X \end{aligned} \qquad (5.3)$$

where $(\vec{X}, j)$ is independent of L.

Often, and especially in the statistical domain, for a given measurement vector $\vec{x}$, one wants to have an estimate of the *probability* that the case is in a certain class j. Thus, one likes to construct estimates for the density

[5] A learning or design sample (domain dependent terminology)

$$p(j|\vec{x}) = p(j|\vec{X} = \vec{x}) \quad j = 1, \cdots, J$$

Hence, instead of constructing classification rules as in equation (5.1), one has

$$\begin{aligned} d&: X \to C^J \\ \vec{x} &\to \vec{d}(\vec{x}) = (\hat{p}(1|\vec{x}), \cdots, \hat{p}(J|\vec{x})) \end{aligned} \tag{5.4}$$

where

$$\forall \vec{x} \in X, \qquad \sum_{j=1}^{J} \hat{p}(j|\vec{x}) = 1 \tag{5.5}$$

These rules are called class probability estimators, [Breiman 1984]. The Bayes estimator is the best achievable estimator. It is given by the true distributions (see section 5.7.1)

$$\vec{d}_B(\vec{x}) = \left(p(1|\vec{x}), \cdots, p(J|\vec{x})\right) \tag{5.6}$$

5.9.2 Overfitting

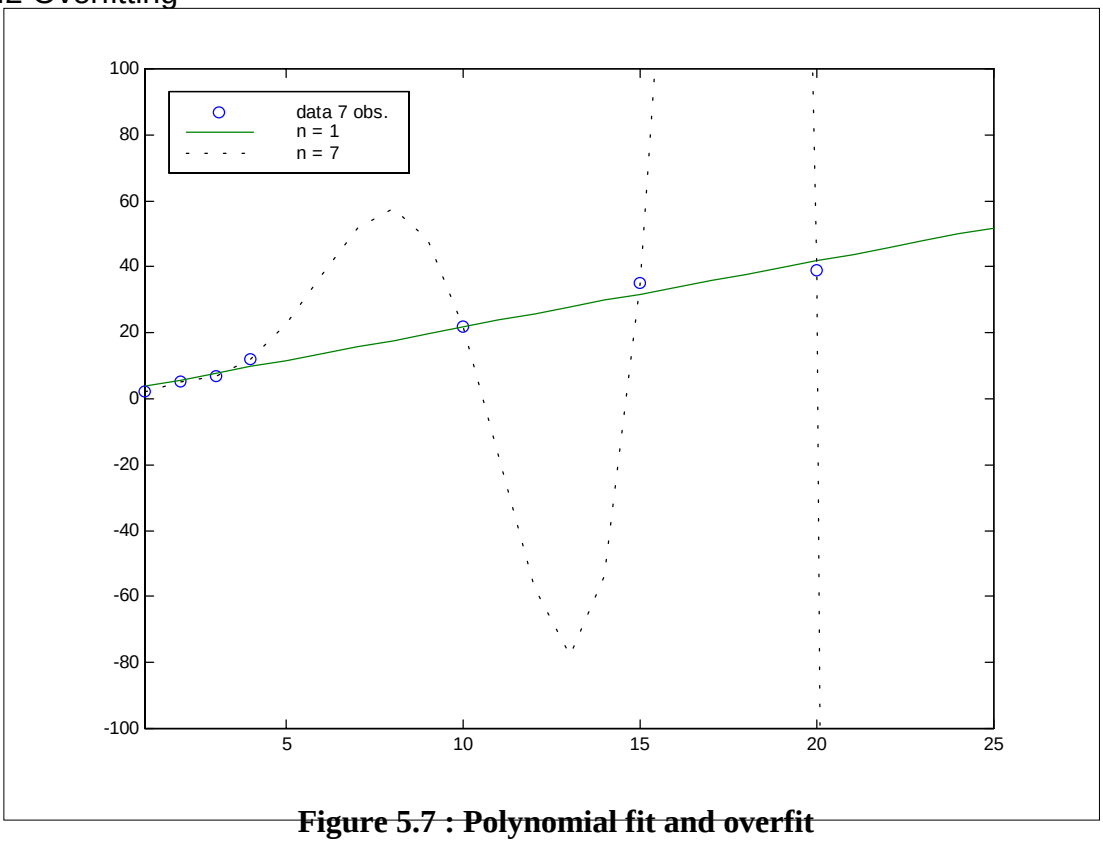


Figure 5.7 : Polynomial fit and overfit

In classification (and regression), one has to select the relevant variables (high predictive power). Selecting too many variables may overfit a model to the data, leading to low errors for the samples from the design set, but high errors on new test samples. This paradigm equally applies to fitting a polynomial on a number of data points: the polynomial will go through all the points if it has an order equal to the number of points (n = 7 in Figure 5.7), but it may be inadequate in interpolation and extrapolation. Hence, a line (n = 1 in Figure 5.7) may be appropriate to acquire a good forecasting behaviour.

Initially, it is often unknown which features have a high quality and which ones are noisy or redundant. Some methods perform well with good predictive features, but they break down when weak features are present. Hence, the goal is to find the best fit to the sample data with-

out overspecialising. Often, an additional parameter is used to measure the complexity fit. The fit with the lowest error in prediction is preferred. It is experienced that in practice simpler classifiers can do a better job than more complex ones (Occam's razor). The example in appendix B illustrates this in section B.3.1.

Overfitting from a statistical viewpoint corresponds with modelling of the unknown idiosyncrasies of the sample that fail to reflect aspects of the true underlying probabilities. Hence, a too complex model may fit the design fit perfectly and thus the underlying true distribution, but it may as well fit the superimposed noise. Models that are more complex will thus introduce additional variance by looking at more variables or by imposing a more complex decision surface (e.g. linear discriminant versus quadratic discriminant analysis). One may conclude that simpler models are more effective in most cases (this is to be compared with Occam's razor).

Minimising the mean squared error can be achieved by minimising the bias and variance. One way to do this is to start with a flexible rule, which will induce a low bias, and then to increase the size of the design set, [Hand 1997]. However, design sets are finite and limited to certain size, thus a compromise between bias and variance has to be found. Hand [1997] describes several strategies for this.

1. The simplest approach is to chose the number of parameters not too large to prevent overfit. In that approach, a design set is used to construct a model, which is then evaluated on a second data set. The model that appears to perform well will be selected. Because the second data set is used to select the best model, it is part of the design of the (near)optimal model: it will also give overly optimistic biased results for it was chosen for that. A third data set is then necessary to measure the model's performance. Changing the number of parameters can be done by forward and/or backward stepwise methods.
2. Another way to obtain a good compromise between bias and variance is the use of a penalty measure. A compromise is sought between accuracy and complexity. For example, a popular penalisation function is the Akaike information criterion (AIC).
3. A third way, which is much related to the second one, seeks a compromise smoothing an overfitted function. Examples are found in the CART methodology, where pruning and averaging tree techniques are used. One can also implement the smoothing of an overfitted function by averaging different predictions from different classifiers.
4. A last method is to smooth the data itself in the hope that this will reduce overfitting problems.

The first method, which relies on a design and test set, is formalised in the next section. This method is used in combination with the third method for tree classifiers. Tree classifiers are discussed in chapter 6. The second method is used in the quality function that SAPS uses (chapter 2).

5.9.3 Error rate and its extensions

If all misclassifications weight equal, then a very simple measure for inaccuracy is error rate. The *empirical* error rate is the ratio of the number of errors to the number of samples, i.e.,

$$\text{error rate} = \frac{\text{number of errors}}{\text{number of cases}} = \text{proportion of misclassifications}$$

The empirical error rate can be read off from a confusion matrix.

The apparent error rate of a classifier is the error rate of the classifier on the sample cases that were used to design or build the classifier, [Weiss 91]. This error rate is also known as the *re-substitution error rate*, [Breiman 84]. Hence, the classification rule has been chosen to optimise some measure of performance relative to the training set.

The *true error rate* is statistically defined as the error rate of the classifier on an asymptotically large number of new cases whose classification has been observed and that converge in the limit to the actual population distribution. These new cases constitute a test sample.

Hence, one has to distinguish between two sets in prediction: the training or design set, and the test or evaluation set. While it is often easy to build classifiers that perform with few or no errors on the original training samples (internal accuracy), it is much harder to have this performance generalised to new test cases (true accuracy). In such situations, it is said that the classifier overfits the training data. It is desirable to have an as reliable as possible estimate of the 'true' error rate, which would be observed on new cases. A corresponding estimator may be biased and thus tend to estimate, or too low (optimistic), or too high (pessimistic). Error-rate estimation techniques can be based on statistical resampling theory, which provides a basis for objectively comparing the error rates of the learning systems on the same data and then estimating their future performance on new data. Additionally, resampling techniques give a sense of the variability in error estimates that one might expect in practice. The only implicit assumption is that the learning samples are at least representative of future samples.

Often, distinctions among different types of error become essential when some predictions that fail are more catastrophic than others are, or occur more frequently than others do. In that case, a confusion matrix is considered indispensable. Correct predictions fall along the diagonal of the confusion matrix, but off-diagonal elements now emphasise the different types of errors with their frequency. If different types of errors have a different impact on the cost of misclassification, then a cost matrix should be used. Taking into account costs, one chooses the decision surface such, that it minimises the expected cost rather that the expected number of misclassifications. Suppose a classifier assigns a class i to a case (i.e., predicts class i) when the actual class was j, then the cost of misclassification, denoted by $C(i|j)$, is defined as

$$\begin{aligned} C(i \mid j) &\geq 0 \qquad i \neq j \\ C(i \mid j) &= 0 \qquad i = j \end{aligned} \tag{5.7}$$

The second equation assumes that a correct classification incurs no cost. A cost for misclassifying a certain class j is defined as

$$\text{cost}(j) = \sum_{i=1}^{n} C(i|j)E(i|j)$$

where the notation $E(i \mid j)$ stands for the number of classifications that are actual of class j, but are classified as class i. $E(i \mid j)$ is found in a confusion matrix.

Special cases arise if one has a constant class conditional cost, or equal costs for misclassifying a class j object, i.e., $C(i|j) = c(1-\delta_{ij})$, or in the unit cost situation (c = 1). Generalisations of cost matrices are found in risk and utility matrices. In risk or decision analysis one also computes the expected gains arising form correct classifications. Utility matrices form an even larger generalisation of risk matrices. In a utility model, risk measures (or costs) are variable.

5.9.4 Formalisation of accuracy for a classifier

The true misclassification error is defined by

$$R^*(d) = P\left(d(\vec{X}) \neq y \mid L\right) \tag{5.8}$$

for an independent sample of the same population. The probability that a case in j is classified in i by a classification rule d in that sample is formally denoted by

$$Q^*(i \mid j) = P(d(\vec{X}) = i \mid Y = j) \tag{5.9}$$

This probability can be estimated the proportion $E(i \mid j)/N$, where N is the total number of classifications. Standard errors can be computed via a binomial probability model. The expected cost of misclassification for class j items is

$$R^*(j) = \sum_i C(i \mid j) Q^*(i \mid j) \tag{5.10}$$

Consequently, the expected misclassification cost for the classifier is given by

$$R^*(d) = \sum_j R^*(j)\pi(j) \tag{5.11}$$

$R^*(d)$ is the proportion of cases in the sample that are misclassified by d. It is given by:

$$R^*(d) = \sum_j \sum_i C(i \mid j) Q^*(i \mid j)\pi(j) \tag{5.12}$$

where the priors are defined by

$$\pi(j) = P(Y = j) \tag{5.13}$$

5.9.5 Internal error estimates

Often, because of limited availability of data, L must be both used to induce a model and to estimate its prediction error. In the case of classification theory, this means that L is used to construct $d(\vec{x})$ and to estimate $R^*(d)$. This is called testing on the training cases. After a model (classifier) is constructed, the cases in L are run through the model. The resubstitution estimate is the most obvious starting point in estimating the true error rate. Formally,

$$\hat{R}(d) = \frac{1}{N} \sum_{n=1}^{N} I_{(d(\vec{x}_n) \neq j_n)} \tag{5.14}$$

where I is the indicator function.

Drawbacks:

- it is the least accurate when tested on new data, because it is based on the design data
- it is computed with the same data used to construct d, instead of an independent sample (clearly a violation of independence).
- hence, it gives an overly optimistic biased picture of the accuracy of d. In other words: it is not an honest estimate.

Overall, it makes no sense to design a classifier that does well on the design samples, but fails on new samples: a resubstitution estimate is likely to overestimate future performance. This is illustrated in Figure 5.8, where the resubstitution error rate $R(d)$ keeps decreasing when used

on the design set. However, use of the classifier on new cases gives another picture for the true error rate $R^*(d)$, which decreases and increase again. Both error rates are high when the complexity of the classifier is very low (underfit). They decrease when the complexity of the classifier increases, but whereas the resubstitution error gives an overly optimistic picture (overfit) for high complex classification models (it keeps decreasing), the true error rate increase again. Hence, an optimum has to be sought. Furthermore, when two classifiers give approximately the same true error rate estimate, the simpler solution is usually taken. The one-standard-error heuristic selects the simplest solution that falls within one error of the minimum error rate. This issue comes back in chapter 6. An illustration of this behaviour can be found in appendix A, section A.4.

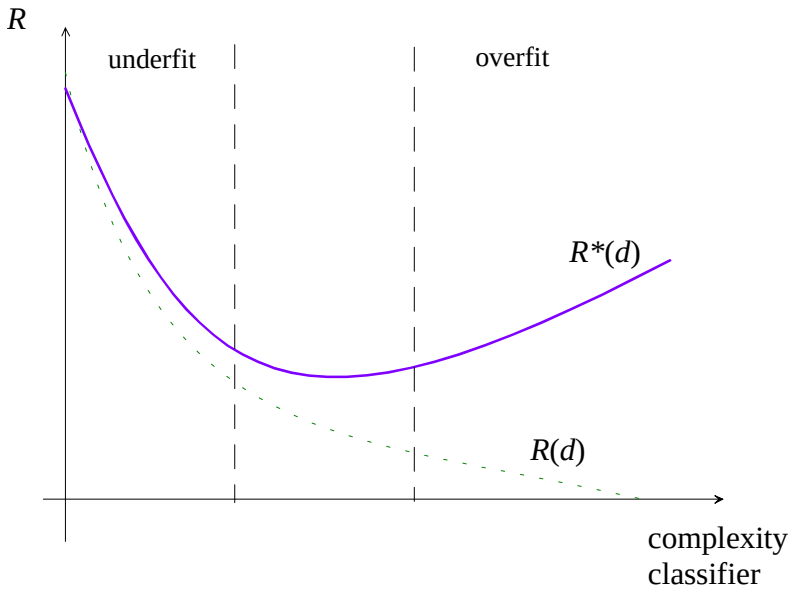


Figure 5.8 : Resubstitution versus true error rate

If the number of sample points would become infinite, the resubstitution estimate converges to the real error rate. Hence, one may wonder how many design cases are needed to be confident at a certain level that the resubsitution estimate is close enough to the true error rate.

Even when an estimate is a very good (unbiased) estimate of the true error rate for a *finite* number of sample points, a simple table lookup as classifier (the observed samples themselves would become the classifier) would still have the problem that a new sample may not have a matching sample in the classifier. It overfits the classifier to the data (modelling noise).

5.9.6 True error estimates

A popular approach to obtain the true misclassification rate is to divide the total number of samples into two groups: a training (design) set and a testing set. The latter constitutes the new cases.

1. A first assumption to be taken is that the design sample data and the training sample data are independent random samples from a certain larger (and same) population, i.e., the samples are representative for the population and the true misclassification rate will be honest. The assumption of a same population may not be always true for, due to evolving things, a population drift may occur. If the population drift can not be modelled, any attempt to have a good classifier performance will fail.
2. Furthermore, the classifications for the design (training) samples are correct (as in ML).

Resampling is a good method to extract the maximum from the data. To get the most out of the data via resampling one has to consider the number of samples available, [Weiss 91].

1. For very large samples size (larger than 1000) the *train-and-test paradigm* can be used. In the context of large databases of data, this method will have our preference.
2. For sample sizes larger than 100 *cross-validation* can be used.
3. For sample sizes less than 100 use leaving-one-out.
4. For very small sample sizes (less than 50) the bootstrap and 2-fold cross-validation can be used in addition to the leave-one-out method.

Methods 2,3 and 4 are resampling methods. Resampling methods (training on the test cases) may provide better estimates of the true error rate, especially when the number of cases is moderate. In resampling many randomly generated train-and-test partitions are used. The estimated error rate is then the average of the error rates obtained with each partition. This method also allows determining standard errors. Estimates obtained via resampling methods are nearly unbiased. In resampling, nearly all the cases are used for training and for testing (be it at different times or in different partitions).

Working under the premise that data sets are large enough (this is the case for all examples in this thesis), a restriction to the first two methods is done.

Train and test paradigm (hold-out method)

The data is divided in two (independent) sets, say L_1 and L_2. L_1 plays the role of training or design set and L_2 acts as testing set. The former is used to construct the classifier (model) and the latter to validate it. Hence, an estimate of the error rate is found by applying the classifier to independent *test* cases. The test sample error-rate is

$$\hat{R}^{ts}(d) = \frac{1}{N^{ts}} \sum_{(\vec{x}_n, j_n) \in L_2} I_{(d(\vec{x}_n) \neq j_n)} \tag{5.15}$$

where the test sample L_2 is composed out of N^{ts} cases.

Under the assumption made at the beginning of this section, one may ask how many test cases are needed for accurate error rate estimation. Regardless of the true population distribution, the accuracy of the error rate estimation is governed by the binomial distribution. Confidence intervals can then be constructed where one notes that the quality (variance) of the test sample estimate directly depends on the number of test cases.

For a classifier *d*, take N_{ij}^{ts} to be the number of class *j* cases in the test sample whose predicted classification by *d* is class *i*. An estimate for the probability that a case with class *j* is classified with class *i* by a classifier *d*, is given by the proportion of test sample class *j* cases that *d* classifies as *i* (see (5.9)):

$$Q^{ts}(i|j) = \frac{N_{ij}^{ts}}{N_j^{ts}} \tag{5.16}$$

where N_j^{ts} cases are in class *j*. In case there are no *i* class cases in the test sample, put $Q^{ts}(i|j) = 0$.

An estimate for the expected cost of misclassification for class *j* items can now be computed (see also (5.10)) and the expected misclassification cost for the classifier is found by using prior estimates (see (5.11)):

$$\hat{R}^{ts}(d) = \frac{1}{N^{ts}} \sum_j \sum_i C(i|j) N_{ij}^{ts} \qquad (5.17)$$

It can be proven that, except for $N_j^{ts} = 0$, the estimates $Q^{ts}(i|j)$ are unbiased. So are the estimates $\hat{R}^{ts}(d)$. In the unit cost case, $\hat{R}^{ts}(d)$ is the proportion of test cases misclassified. Hence, the standard error on $\hat{R}^{ts}(d)$ is given by

$$se(\hat{R}^{ts}(d)) = \sqrt{\frac{\hat{R}^{ts}(d)(1-\hat{R}^{ts}(d))}{N^{ts}}} \qquad (5.18)$$

Remark that this standard error decreases with increasing size of the test set. When the number of test samples equals 1000, the standard error is certainly below 2 % (in the worst case where $\hat{R}^{ts}(d) = 1/2$).

Usually, the training set contains 2/3 or 1/2 of the data, while the testing set contains the remaining part of the data (holdout method). When there are at least 1000 test cases available, larger percentages for training can be retained.

Advantages and drawbacks

+ it is computationally efficient
+ it is preferable when there are many cases available
+ it is virtually unbiased (except when some classes are missing).
+ it is philosophically a sound concept (truly new cases are evaluated)
– it reduces the effective sample size, which may be especially a problem in small data sets.
– information contained in the test set is not used to construct the model.

K-fold cross-validation

K-fold validation is a resampling method. For K-fold cross-validation, the original learning (training) sample is divided by random selection into *K* subsets, and *K* auxiliary classifiers are constructed on *L*. The k^{th} classifier is constructed based upon the learning sample $L^{(-k)} = L \setminus L_k$ (= $L \cap \text{co}(L_k)$) and the corresponding test set is L_k. Hence, K-fold cross-validation is a way of resampling. Algorithmically, it can be stated as:

1. Split the data into *K* roughly equal parts
2. For the k^{th} part (L_k plays the role of test set), fit the model to the other *K*-1 parts of the data ($L^{(-k)} = L \setminus L_k$ plays the role of training set), and compute the misclassification rate or prediction error of the fitted model when predicting or classifying the k^{th} part of the data.
3. Do the above for $k = 1,2, \ldots, K$ mutually exclusive test partitions and gather the *K* estimates of misclassification or prediction error (rotate the roles).
4. The average error of all *K* partitions is the cross-validated error rate.

Formally, denote $d^{(-k)}$ as the classifier for the data set without part k. $\hat{R}^{ts}(d^{(-k)})$ is a test sample estimate for $R^*(d^{(-k)})$, then (step 2)

$$\hat{R}^{ts}(d^{(-k)}) = \frac{1}{\#(L \setminus L_k)} \sum_{(\vec{x}_n, j_n) \in L \setminus L_k} I_{(d^{(-k)}(\vec{x}_n) \neq j_n)} \tag{5.19}$$

Hence, each of the classifiers $d^{(-k)}$ (step 3) is using almost all information in L and has a misclassification rate nearly equal to $R^*(d)$. The K-fold estimate is then defined to be (step 4)

$$\hat{R}^{CV}(d) = \frac{1}{K} \sum_{k=1}^{K} R^{ts}(d^{(-k)}) \tag{5.20}$$

Due to the computational expensiveness of the method, it is mainly used when sample size is relatively small.

To compute the estimated expected misclassification cost, one proceeds as in the train-and test paradigm. First, define $N_{ij}^{(k)}$ as the number of class j cases in the test sample L_k classified as i by $d^{(-k)}$. The total number of class j test cases classified as i is then

$$N_{ij} = \sum_k N_{ij}^{(k)} \, . \tag{5.21}$$

As each case appears in only one test sample, the total number of class j cases is the number of class j cases in L, i.e. N_j.

If K is sufficiently large, $d^{(-k)}$ should have about the same classification accuracy as d. Thus, one assumes that the estimated misclassification error probability is given by

$$Q^{CV}(i|j) = \frac{N_{ij}}{N_j} \tag{5.22}$$

Working analogous as in the train-and test paradigm, one finds, when using or estimating prior estimates that

$$\hat{R}^{CV}(d) = \frac{1}{N} \sum_j \sum_i C(i|j) N_{ij} \tag{5.23}$$

The computation of the standard error on the test cases, via an analogous equation as in the train-and test paradigm, is a bit optimistic because the test cases are not completely independent anymore.

Advantages and drawbacks of cross-validation

+ It is quite general and it gives similar answers as standard methods in simple problems
+ It is robust; it works also when the model is not entirely correct. It uses all cases and provides information regarding the stability of the tree structure.
+ Simulation studies shows it is nearly unbiased but has a large variability for small samples
− The use of cross-validation is computationally less efficient

Remark:

In both the test sample and the cross-validation approach, selections are done in L to achieve the test samples or cross-validation test samples. One can randomly divide L in the respective sets but this may lead to random class representations in the samples. Constructing test sam-

ple(s) to contain fixed fractions of the classes may produce more accurate estimates [Breiman 1984].

5.10 Conclusion

It has been demonstrated that data-mining is an essential step in the overall process of KDD. The knowledge-discovery approach is compared with inductive learning. Supervised learning, a term borrowed from the domain of machine learning, will prove to be the underlying paradigm for SAPS. Classification and regression are methods that belong to the supervised learning paradigm, which itself is an answer to the 'Classification and Prediction' problem type. However, a restriction to classifiers is done because the tree classifier approach, which is the main topic of chapter 6, form a natural stepping stone to regression trees [Breiman 1984].

Classifiers all do a successive partitioning of a feature space of predictors into subsets, where the partition is done on a design sample and the validation on a test sample. The hope is to partition the feature space such that the resulting regions are simple enough to be understandable and yet as homogeneous as possible with regard to the outcome. Computerised classification methods are much faster than humans and can process much more information. In classification, one usually works with crisp classes where each class corresponds with an outcome. Furthermore, the usual restriction to attribute-value decomposition is applied. Accuracy of a classifier is formalised, and the resubstitution error rate versus the true error rate shows that internal validation gives optimistic biased error rate estimations. The latter applies for SAPS: it is illustrated by the example in appendix B.

Resubstitution error estimates can be seen as a measure for replicatively validity in chapter 1, while true error estimates can be seen as a measure for predictively validity.

Finally, this chapter explained the train-and-test paradigm and the use of cross-validation in classification. The one-standard-error heuristic is described and the typical form of the true error estimate is explained.

References

Agrawal R., Psaila G., Wimmers E. L., and Zait M. [1995], "*Querying shapes of histories*", VLDB'95, p. 502-514, Zurich, Switzerland, September 1995.

Agrawal R., Mannila H., Srikant R, Toivonen H., Verkamo I. [1996], "*Fast Discovery of Association Rules*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 307-328, 1996.

Berndt D. J., Clifford J. [1996], "*Finding Patterns in Time Series: A Dynamic Programming Approach*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 229-248, 1996.

Berry M. J. A., Linoff G. [1997], Data Mining Techniques For Marketing, Sales and Customer Support, Wiley Computer Publishing, 1997.

Brachman R. J., Anand T. [1996], "*The Process of Knowledge Discovery in Databases*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 37-58, 1996.

Breiman L., Friedman J. H., Olshen R. A., Stone C. J. [1984], Classification and Regression Trees, Chapman & Hall, 1984.

Faloutsos C., Ranganathan M., and Manolopoulos Y. [1994], "*Fast subsequence matching in time-series databases*", Proceedings ACM SIGMOD, Minneapolis MN, May 25-27, p. 419-429, 1994

Fayyad U. M., Piatetsky-Shapiro G., Smyth P. [1996], "*From Data Mining to Knowledge Discovery: An Overview*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 1-34, 1996.

Feng C., Michie D., [1994] "*Machine Learning of Rules and Trees*", Machine Learning, Neural and Statistical Classification, ed. Michie D., Spiegelhalter D. J. and Taylor C.C., Ellis Horwood, chapter 5, p. 50-83, 1994.

Gaines B. R. [1996], "*Transforming Rules and Trees into Comprehensible Knowledge Structures*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 205-226, 1996.

Guyon I., Matic N., Vapnik V. [1996], "*Discovering Informative Patterns and Data Cleaning*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 181-204, 1996.

Han J. [1999], personal communication, Tutorial given in UIA in April 1999.

Hand D. J. [1997], Construction and Assessment of Classification Rules. John Wiley & Sons, 1997.

Heckerman D. [1996], "*Bayesian Networks for Knowledge Discovery*" Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 273-306, 1996.

Henery R.J. [1994], "*Classification*", Machine Learning, Neural and Statistical Classification, ed. Michie D., Spiegelhalter D. J. and Taylor C.C. , Ellis Horwood, chapter 2, p. 6-16, 1994.

Information Discovery Inc. [1998], "*A Characterization of Data Mining Technologies and Processes*", on the web page www.datamining.com/datamine/techwp.htm

Klösgen W. [1996], "*Explora, A Multipattern and Multistrategy Discovery Assistant*" Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 249-271, 1996.

Michalski, R.S., Chilausky R. L. [1980], "*Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis*", International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2, p. 125-161, 1980.

Michie D., Spiegelhalter D.J., Taylor C.C. [1994], Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994.

Mitchell T.M. [1997], Machine Learning. McGraw-Hill, 1997.

Quinlan J.R. [1993], C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning, 1993.

Salzberg S. L. [1997], "*On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach*", Data Mining and Knowledge Discovery, 1, p. 317-328, 1997.

SAS Institute [1996], "*Data Mining with the SAS System: From Data to Business Advantage*" (SAS Institute White Paper), 1996, see also www.sas.com

Siftware [1999], See www.kdnuggets.com/tools.html

Simoudis E., Livezey B., Kerber R. [1996], "*Integrating Inductive and Deductive Reasoning for Data Mining*", Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 353 - 373, 1996.

Think [1999], See www.think.com

Thornton C. J. [1992], Techniques in Computational Learning; An Introduction. Chapman & Hall, 1992.

Van Welden D., Vansteenkiste G.C. [1996], "*Sub-Optimal Mask Search in SAPS*", International Journal of General Systems, vol. 24, 1-2, p. 137-150, 1996.

Van Welden D. [1998], Tree Classifiers as Data Mining Tools. MSc. Thesis, Catholic University of Leuven, Belgium, 1998.

Weiss S. M., Kulikowski C. A. [1991], Computer Systems That Learn. Morgan Kaufmann Publishers, 1991.

Chapter 6

Classification and Regression Trees

6.1 Introduction

Tree classifiers are a special kind of classifiers that rely on a hierarchical recursive partitioning scheme. Tree induction starts with a root node to create a tree of depth 1 by partitioning the training set among the leaves just created. Then, the same algorithm is applied to each leaf until some criterion is met. The leaves form the final partition of the measurement space. A tree classifier classifies starting at the root, moving down and performing a test (decision) at each node until a leaf node is reached. The unique path from the root to each leaf[1] can be considered as a rule classifying the records. Depending on the nature of the response, tree classifiers are referred to as classification trees (discrete response), regression trees (continuous response), survival trees (censored positive response), or tree-structured vector quantisation (when the predictors and response are the same), see [Armitage 1997]. Only classification and regression trees are considered in this chapter.

Induction techniques for tree classifiers have been developed in parallel both within the machine learning community, motivated by knowledge acquisition for expert systems, and within the statistical community as a response to the perceived limitations of classical linear discrimination techniques. Hence, two representative classifiers, C4.5 and CART[2], will illustrate the tree classifier approach. Regression trees are very similar to classification trees: the latter are a kind of logical prerequisite for the former. Hence, classification trees will be discussed in more depth and only the changes that apply for regression trees are highlighted at the end of the chapter.

6.2 Classification tree examples

Classification trees can be classified into decision trees (a leaf contains only one class) or class probability trees (a leaf contains a class distribution). However, this distinction is not always made and pruning can convert a decision tree into a class probability tree. Decision trees can be considered as a special case of class probability trees, which are more realistic in the presence of noise.

C4.5© is the representative classification tree method in the machine learning domain. It has been devised by Quinlan [1993]. It typically uses a standard structure and the goal attribute is usually categorical. Non-goal attributes or predictors can be any type. C4.5 is briefly described in section 6.8.2. Meanwhile, a successor, called See5, entered the market [Rulequest

[1] A leaf is also called a terminal node.

[2] CART in an acronym for Classification And Regression Trees, but it is as well a software package that does classification and Regression. The abbreviation refers to the software package.

1999]. The representative classification and regression methods in the domain of statistics are implemented in a software package, named CART© [Salford 1999]. It is based on [Breiman et al. 1984].

An example, which is situated in the domain of statistics and which originates from [Breiman et al. 1984], demonstrates some basic properties of classification trees. The example describes the problem of early detection of hart attack risk patients on the basis of 19 measured variables (blood pressure, age, …) for the first 24 hour data of their arrival. As in C4.5, predictors can be any type. The classes, one is interested in, are low-risk or high-risk patient. The induced tree only retains only 3 from the 19 variables; the others appear to be irrelevant. This leads to the conclusion that if a variable is never split on in the final tree, one may assume it has no importance in determination of class membership. Consequently, tree classifiers can be used for relevance analysis or feature selection. One should be careful though, because a variable's effect may be masked by other variables. A method for ranking variables in terms of their potential effect on the classification may indicate if masking effects are present or not. More information about ranking of variables is found in section 6.9.4.

Tree classifiers that use univariate splits have a simple interpretation if the trees are not too large. A monothetic tree classifier divides the measurement space by creating with each division a hyperplane orthogonal to the axis of the tested attribute, and which is parallel to all other axes. The regions thus produced by such divisions are hyperrectangles. When the problem is such that the class regions can not very well be represented by hyperrectangles, the best a monothetic (or orthogonal) tree classifier can do is to approximate the regions by hyperrectangles. This is illustrated in Figure 6.1. Despite this extra inductive bias to hyperrectangles, orthogonal splits are not that restrictive (with regard to accuracy) as it may seem from first sight, and as can be noticed from Figure 6.1.

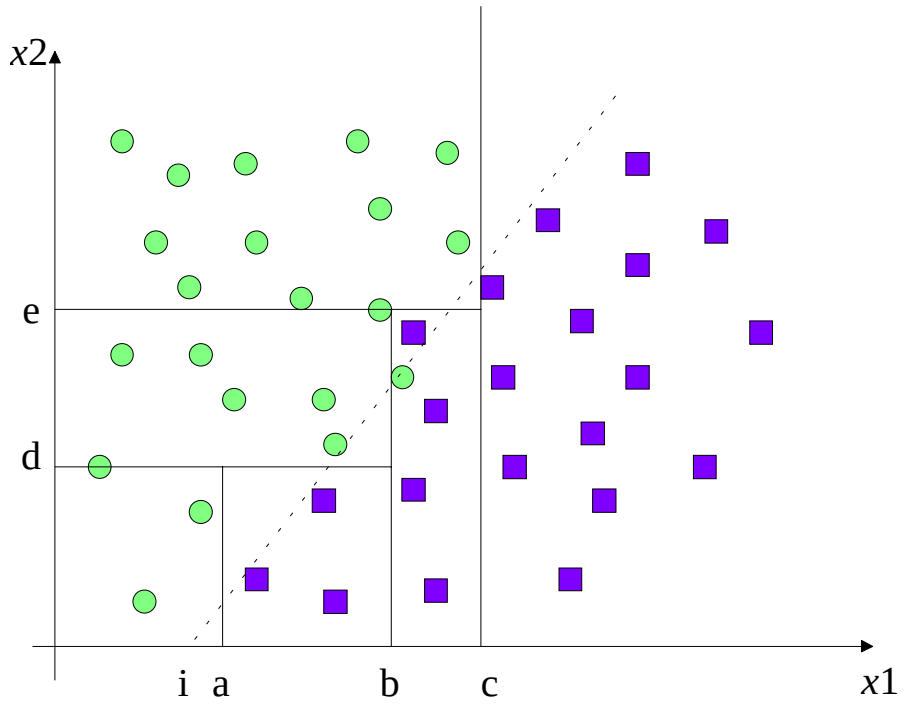


Figure 6.1 : Example of divisions for monothetic tree classifiers (full lines)

Suppose that in Figure 6.1 the circles belong to class ‘circle’, and the squares to class ‘square’, then a monothetic division looks like Figure 6.2 (remark the re-use of a same variable for splitting).

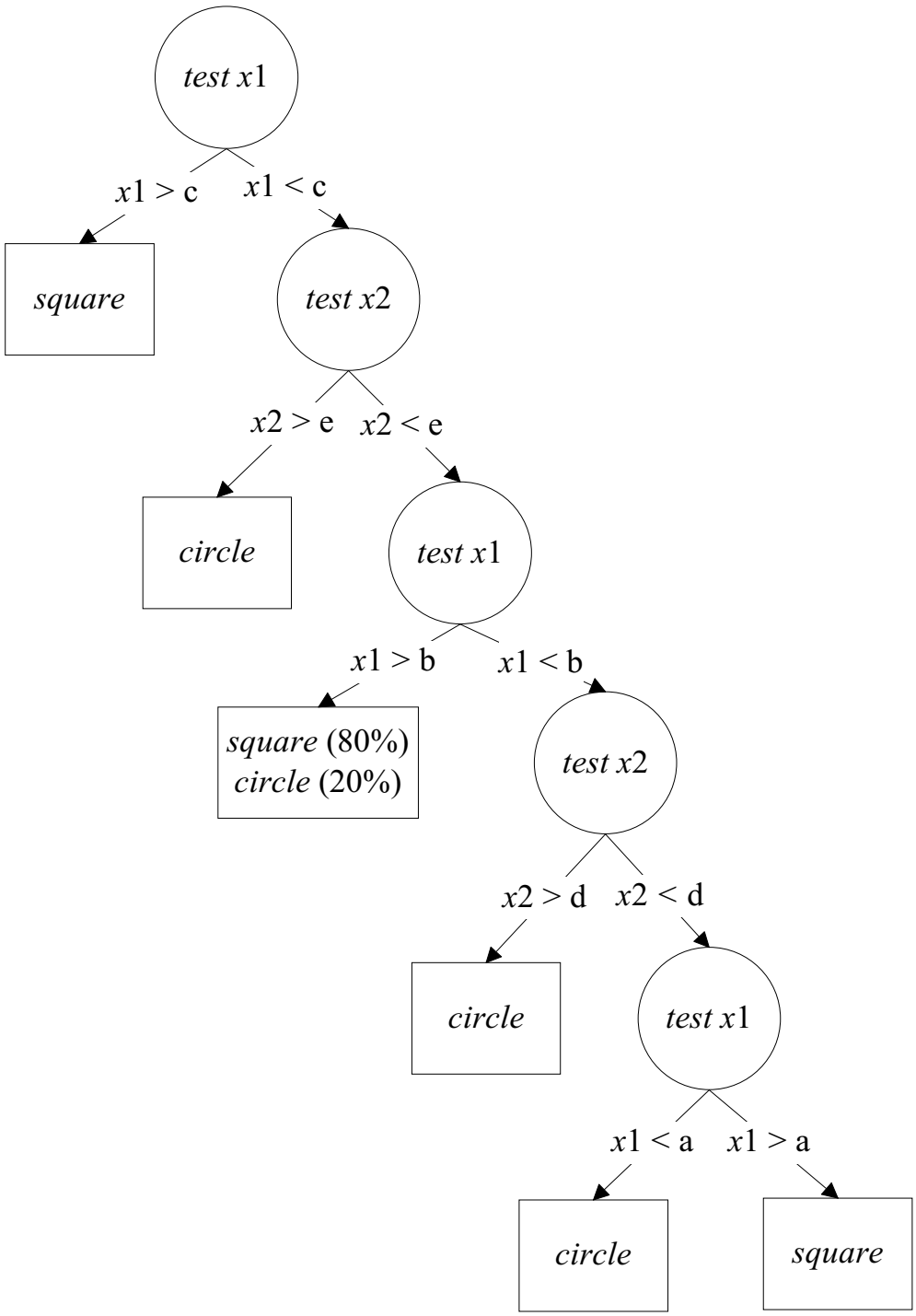


Figure 6.2 : Example of monothetic tree classifier

A classification tree can be represented in conjunctive-disjunctive form: all attribute-name nodes become disjunctive relations and all attribute-value nodes become conjunctive relations. The decision tree is in fact an AND-OR tree. This is shown in Figure 6.3.

```
if      x1 > c then 'square'
else
        if      x2 > e then 'circle'
        else
                if      x1 > b then 'square' (80%) or 'circle' (20%)
                else
                        if      x2 > d then 'circle'
                        else
                                if      x1 > a then 'square'
                                else    'circle'
```

Figure 6.3 : Example of rules obtained by a monothetic tree classifier

As the number of training cases increases, the approximation of an oblique division by a collection of hyperrectangles becomes better. However, this is at the expense of a serious increase in the number of smaller regions, which renders the interpretation more difficult. If an unlimited growth of the tree occurs as more training cases are used, coupled with a more or less constant error rate on the same training cases, then this is an indication that orthogonal hyperplanes do not perform well.

The use of threshold splits parallel to the feature axes limits the type of classification boundaries that can be induced (e.g., where the class structure depends on combinations of variables). Figure 6.4 illustrates Breiman's statement that oblique trees are competitive or better than linear discriminant analysis, [Breiman et al. 1984].

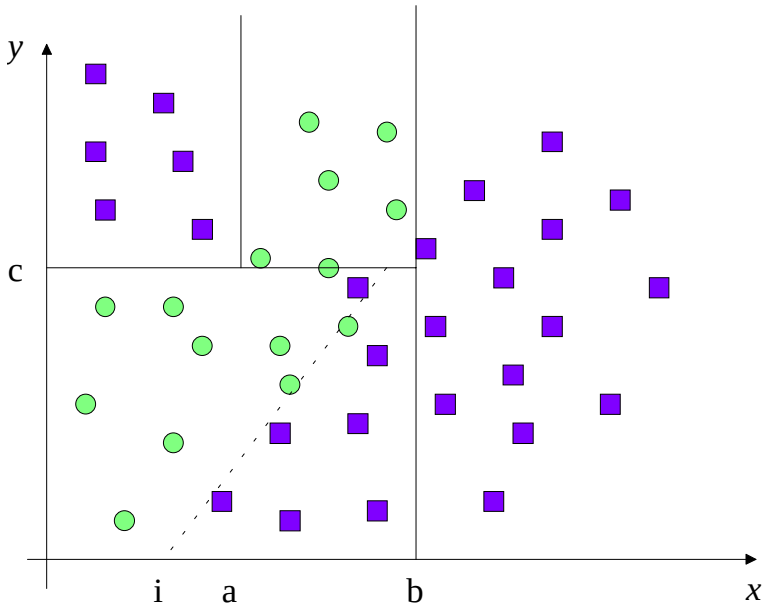


Figure 6.4 : A more complex partitioning

For the example in Figure 6.1, the equation for the dashed line becomes

$$x_2 - x_1 - i > 0$$

and the other (orthogonal on the axes) lines are the instantiations

$$x_1 - c > 0, x_2 - e > 0, x_1 - b > 0, x_2 - d > 0, x_1 - a > 0$$

It is obvious that oblique trees are more efficient and accurate when the concepts to recognise are defined by a polygonal partition of the measurement space. This partition is easier approximated by non-linear hypersurfaces, but is computationally more complex. This is why, after the pioneering work of Breiman, one had to wait for the 90s to obtain practical methods for the induction of oblique trees. Such methods are explained in [Murthy 1997].

Model space enlargement increases the model power, e.g. via multivariate hyperplanes at arbitrary angles, but decreases the comprehensibility of the model. Consequently, one could ask whether the partition of each node should be based on just one variable (univariate or monothetic) or on a combination of variables (multivariate or polythetic). Some people reject the use of oblique trees because they also may become uninterpretable [Kodratoff 1997]. This is a personal viewpoint of some authors, for in principal component analysis a combination of variables is taken too to describe variance components. A sensible approach is to make use of (whatever) meaningful combinations of variables that facilitate the induction. In this way, a kind of model is set forward (like in factor analysis) and the tree searches the best splits. As a result, the interpretability is enhanced and the power of the model is increased.

6.3 Tree induction principles

At each node in a tree, different things can be measured to assist its construction:

- the number of cases in a node and the distribution of the classes in a node,
- the classification done at a node if the node is considered a leaf, and how well the classification performs (error rate, cost of misclassification, etc.).

In the actual construction of a tree, three basic questions have to be answered:

1. How to split a node? This is elaborated in section 6.4.
   a) What is the splitting rule? This is related with the distribution of classes in nodes.
   b) What is the arity of the splitting?
2. When to declare a node terminal, i.e., what is the stopping rule? A stopping rule can be based on the distribution of classes in a node; on the number of cases left in the node; and on how good the classification performs. This forms the topic of section 6.5.
3. What inferences to make for the various (uniform, not-uniform) terminal (leaf) nodes, i.e., what is the assignment rule? This is of course related to how the classification is done at a leaf. This is described in section 6.6.

6.4 How to split a node: splitting rule?

The purpose of splitting a node is to generate offspring who are more preferred than the root node in some sense. Desirable splits are the ones for which the distributions of the outcome in the child nodes are more homogeneous (purer) than in the root node. This is depicted in Figure 6.5. Put in other words: the measure to evaluate a potential splitter is diversity, [Berry 1997]. It can be formalised by an impurity function. The impurity in a node *t* is formally denoted by *i*(*t*).

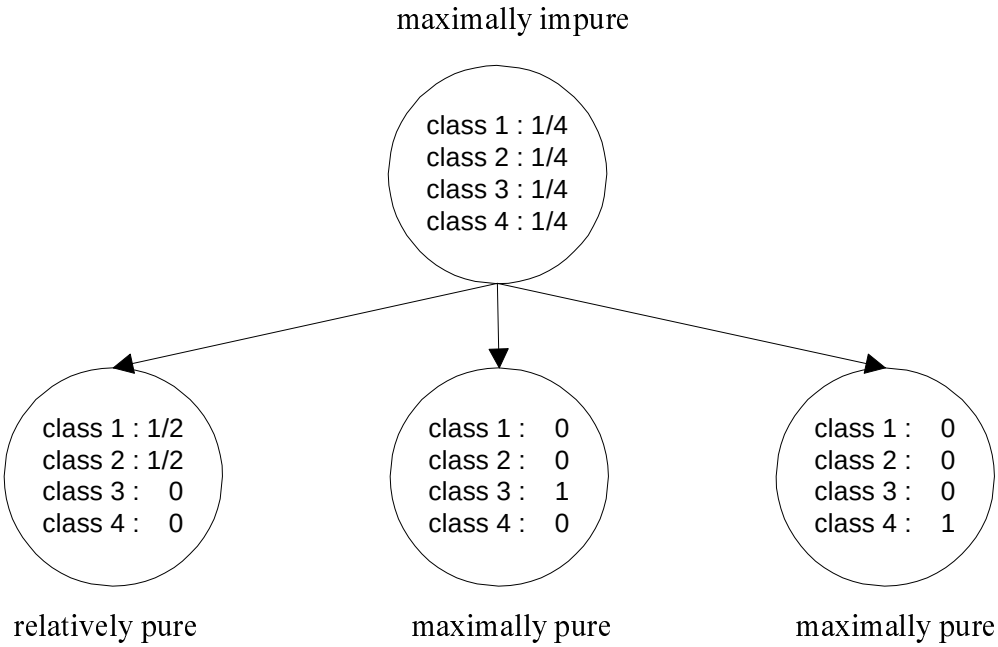


Figure 6.5 : Example of purity in nodes

Starting at the root node, each node is split with the help of a splitting criterion. For a given splitting criterion, one could try to find directly the smallest decision tree consistent with a training set. This problem is, however, NP-complete, [Hyafil 1976]. Consequently, most tree induction methods use greedy algorithms and, hence, do not backtrack. A heuristic that evaluates impurity (reductions) is used to select the next most appropriate split. This implies that a test, which is used at a node to partition the training cases, must be evaluated without exploring subsequent divisions lower in the tree. Consequently, the only information available for guidance is the distribution of classes in the training set, in the parent node, and in the child nodes (along with the respective number of cases). The goodness of a split *s* in a node *t* is defined to be a weighted decrease in impurity, i.e.,

$$\Delta i(s,t) = i(t) - \sum_{descendants} p_{descendant|s} . i(descendant|s) \qquad (6.1)$$

where $p_{descendant|s}$ is the proportion of cases that go into the corresponding descendant node for a given split *s*. Impurity of a tree *T* can be defined as the sum of the (weighted) impurities of the terminal nodes, where the weighted node impurity in a node *t* with probability of occurrence of $p(t)$ is given by $I(t) = p(t)i(t)$, where $p(t)$ is the probability of occurrence of the node *t*. In the case of a binary tree, the set of questions is binary and some things simplify. The words 'left' and 'right' descendants make sense now. By convention, cases answering "yes" go to the left descendant node, those answering "no" go to the right descending node.

At each node *t*, all candidate splits *s* are explored. The split s^* that gives the largest decrease in impurity is then applied, i.e.,

$$\Delta i(s^*, t) = \max_{s \in S} \Delta i(s, t) \qquad (6.2)$$

where *S* is the set of all possible splits at node *t*.

The greedy algorithm looks only one step ahead, but it has a major drawback, which can be seen by a checkerboard configuration. This configuration is an example for which no impurity function can be found that results in a good splitting. Therefore, tree classifiers are easily stuck in local minima (some other methods such as genetic algorithms are not, but this is outside the scope of this thesis). Another disadvantage of any splitting rule is that it is subjective to the sampling variation. The influence of the sampling variation is more pronounced when there are fewer cases in a node. Thus, especially at the bottom of a tree classifier, splits are less reliable than at the top. This, of course, is linked with overfitting. Pruning back an overfitted tree solves this problem to a great extent, see section 6.5 and section 6.9.

To compute the decrease in impurity, a specific impurity function has to be agreed on. This thesis is limited to the case of standard structure data.

6.4.1 The relation between possible splits and variable type

For nominal variables, the test at each node takes the form $x \in \wp(\{b_1, \cdots, b_L\})$ ($\wp(A)$ is the level power set of *A*), where the set inside the brackets is the set of values *x* can take (levels of *x*). Hence, this may yield soon enough too many splitting possibilities to be explored exhaustively.

From ordinal variables on, one cannot amalgamate levels together just like that. The ordering has to be respected[3]. In C4.5 and CART, only binary splits are considered. Then, for *n* ordered levels of a variable, only *n*-1 possible splits are considered.

For a continuous variable, there is normally no 'space' between the levels to split. However, each variable is quantised when measured and stored. Sorting a variable on its values will result in a finite (perhaps very large) set of values. A split can occur between any of these values, and the situation for an ordered set of levels is applicable. Thus, for the levels $\{v_1, v_2, \cdots, v_n\}$ one can split at each midpoint of an interval of two adjacent values, which results in a threshold between classes of the form $(v_i + v_{i+1})/2$. The appearance of continuous variables may thus impose a heavy load on computation time. Fortunately, efficient algorithms are devised that alleviate this problem, [Quinlan 1993]. Oblique trees come into play when continuous variables are present. The test at each node can then take the form of a linear combination approach, i.e.,

$$\sum_{i=1}^{n} a_i x_i - c_{threshold} > 0 \qquad \sum_{i=1}^{n} a_i^2 = 1$$

where $a_1, \cdots, a_n$, and $c_{threshold}$ are real valued coefficients (or weights as in [Quinlan 1993]) that are determined such as to maximise the performance of a split. Nominal variables are excluded for linear combination, but the form can be modified to included Boolean variable combinations via conjunction and/or disjunction.

Whatever the type of variable, if one can choose appropriate groups of attribute levels based on domain knowledge, then this should be done. It reduces computational complexity and it

[3] Not always: what about extreme and medium (author's opinion)?

may give a more natural interpretation while giving sufficient data to split further than in the non-grouped case. This kind of grouping can hardly be automated, except perhaps with some help from an expert system that knows about variables and their meaning. For example, when a variable is water temperature and the range is from - 10 degrees Celsius to + 200 degrees Celsius, then a grouping may be based on the state of the matter: ice (solid), liquid, and vapour (gas).

6.4.2 Node probability determination

In the definition of impurity measures, conditional probabilities are used. Often, frequency estimates are needed for the corresponding probabilities. For the root node, one often takes as prior probabilities (priors), denoted by $\pi(j)$, the proportions N_j/N (frequency approach), where N_j is the number of cases in class j and N the total number of cases, i.e.,

$$\pi(j) = \frac{N_j}{N} \tag{6.3}$$

Another possibility is to let experts fix these numbers. $\pi(j)$ is interpreted as the probability that a class j will be presented to the tree. It may not necessarily reflect proportions to be expected in future cases because the data may be filtered in one way of another.

A node t has a probability of occurrence of $p(t)$. If a node t has $N(t)$ cases, for which $N_j(t)$ is the number of cases that correspond with class j, the estimate for the conditional probability $p(j|t)$ is estimated by the class proportions for a node t, i.e.,

$$p(j|t) = \frac{p(j,t)}{p(t)} = \frac{N_j(t)}{N(t)} \tag{6.4}$$

$p(t)$ can also be estimated immediately by

$$p(t) = \frac{N(t)}{N} \tag{6.5}$$

Hence,

$$p(j,t) = p(j|t)\,p(t) = \frac{N_j(t)}{N} = p(t|j)\pi(j) = \frac{N_j(t)}{N_j}\pi(j) \tag{6.6}$$

Denote the set of terminal nodes or leafs by $\tilde{T}$, then $\sum_{t\in\tilde{T}} p(t) = 1$.

Contingency tables aid in the determination of impurity reductions. They can be represented in a joint distribution form or in a row-conditional form. The latter is the most logical one because the classes form the response.

For a given (parent) node t with J classes, a test X or a split s that results in splitting on a certain attribute may produce, say, I child nodes labelled $t_1, t_2, \cdots, t_I$. The classes are labelled by $1, 2, \cdots, J$. A row-conditional contingency table, which corresponds with a generalisation of Figure 6.5, is depicted in Table 6.1.

Child\class	class 1	…	class j	…	class J	
Child t_1	$p(1\mid t_1)$		$p(j\mid t_1)$		$p(J\mid t_1)$	1
…						
Child t_i	$p(1\mid t_i)$		$p(j\mid t_i)$		$p(J\mid t_i)$	1
…						
Child t_I	$p(1\mid t_I)$		$p(j\mid t_I)$		$p(J\mid t_J)$	1

Table 6.1 : Contingency table for splitting in row conditional form

6.4.3 Measures of impurity

A measure of impurity for a node t is a non-negative function ϕ of the node conditional probabilities $p(j|t)$, i.e., $i(t) = \phi(p(1|t), p(2|t), \cdots, p(J|t))$, such that

- it is maximal if these probabilities are equal, i.e.

$$\phi\left(\frac{1}{J}, \frac{1}{J}, \cdots, \frac{1}{J}\right) \text{ is maximum}$$

- it is zero if one probability is 1 and the others zero, i.e.

$$\phi(1,0,\cdots,0) = \phi(0,1,\cdots,0) = \cdots = \phi(0,0,\cdots,1) = 0$$

- it is symmetrical, i.e.

$$\phi(\Gamma(p(1|t), p(2|t), \cdots, p(J|t))) = \phi(p(1|t), p(2|t), \cdots, p(J|t))$$

with Γ a permutation operator.

Hence, node impurity is maximal when all classes are equally mixed, and smallest when the node contains only one class. To reward purer nodes enough, one has to add a strictly concavity condition, i.e.,

$$\phi''(p(1|t), \cdots, p(J|t)) < 0$$

see [Breiman et al. 1984; Van Welden 1998]. A fundamental property of such impurity functions is that for any node t and split s, one has that $\Delta i(s,t) \geq 0$. A proof, based on a property of convex functions, can be found in [Van Welden 1998]. Another reason to select this class of impurity function is found in [Breiman et al. 1984], where the authors note an important computational advantage for nominal variables. A disadvantage is that such impurity functions are not good indicators for stopping further splitting. Hence, backward pruning should be applied when using such a criterion.

A good choice for impurity is based on a quadratic polynomial function, which fulfils the above requirements for a impurity function for the two-class problem. Extension to multiple classes results in the Gini-index, which is defined as

$$i(t) = \sum_{i=1}^{J} \sum_{\substack{j=1 \\ j \neq i}}^{J} p(j\mid t)\, p(i\mid t) = 1 - \sum_{j=1}^{J} p^2(j\mid t) \qquad (6.7)$$

The Gini-index can be interpreted as an estimated probability of misclassification under random assignment, or as a variance for dichotomous classification.

Another good choice for impurity measure relies on the utilisation of the Shannon entropy. It is given by

$$i(t) = -\sum_{j=1}^{J} p(j \mid t) \log_2 p(j \mid t) \qquad (6.8)$$

Impurity (in a node *t*) is often associated with the average amount of information needed to identify the class of a case in a given set cases (in the node *t*). Consequently, it is often denoted by info(*t*). A node *t* can be regarded as an instantiation of a random number *Y*. *Y* has then as (finite) population all possible nodes in a tree.

$$H(Y) = -\sum_{j=1}^{J} p_Y(j) \log_2 p_Y(j) = \text{info}(Y) \qquad (6.9)$$

where $p_Y(j)$ is the density distribution in a node *Y*.

Each child node *Y* in a split corresponds with an element in the level power set of an attribute *X* for its parent node, see section 6.4.1. The acceptable partitions on this set form a population of possible splits. For a given partitioning, the representative components[4] can be labelled to a finite set of integers, e.g., {1,…,*I*}. Consequently, $X = i$ denotes a certain choice from a test or split. It corresponds with a certain child node for which the value of the attribute *X* is known. Hence, one can determine the information in a child node by the conditional entropy of *Y* given $X = i$, i.e.,

$$H(Y \mid X = i) = \text{info}_{X=i}(Y) = \text{info}(Y \mid X = i) \qquad (6.10)$$

The average child info is then given by the conditional entropy of *Y* given *X*, i.e.,

$$H(Y \mid X) = \sum_{i=1}^{I} p_X(i) H(Y \mid X = i) \qquad (6.11)$$

Equation (6.11) corresponds with equation (2.7). The average reduction in uncertainty about *Y* that results from learning the value of *X* can then be expressed by the mutual information between *X* and *Y*, i.e., $H(Y; X)$.

Many other rules can be devised, see [Breiman et al. 1984; Mingers 1989], but only the Gini rule and the Shannon rule are further necessary in this thesis. The former is used implicitly in the appendix examples, while the latter shows a remarkable resemblance with SAPS (see chapter 2).

6.5 How to declare a node terminal: stopping rule?

In this section, two strategies for deriving stopping rules are described and explained. It is known that the splitting process has to terminate because the number of cases is finite. Regardless that, continuing splitting too much would yield a generally too large tree to be useful (overfitting) and this not only with regard to the comprehensibility, but also with regard to classification errors. A pruning strategy has to be designed to prevent overfitting because a tree classifier is important, not because it summarises what is known (the training set), but because it hopefully classifies new cases correctly (see section 5.10.2).

[4] The term equivalence class might confuse the reader in this context.

There are essentially two pruning techniques available:

1. forward pruning
2. backward pruning

Deepening a tree can be stopped when the purity measure exceeds a certain threshold. However, this rule may be never fired if, e.g., a node cannot be made purer anymore. Setting the threshold too low may lead to too much splitting (large trees) and setting it too high may lead to premature stopping although further splitting of descendant nodes could give a valuable increase in purity even above the threshold.

A more practical and yet simple rule for determining if a node is a leaf, could be to look if a significant *decrease* in impurity is still possible, see equation (6.2). This is done by setting a threshold on the decrease in impurity. A node is then declared terminal (a leaf) if the maximum decrease in impurity in a node is smaller than the chosen threshold β, i.e.,

$$\max_{s \in S} \Delta i(s,t) < \beta$$

This rule is called the decrement threshold rule.

The above stopping rules are part of an approach called forward pruning. In that case, leaves may no longer contain pure nodes, but instead a frequency distribution over classes, i.e., one has class probability trees.

Often, another rule is added to prevent overfitting: one may split further and further until one arrives at single-class subsets, even if all or most of them contain a single training case. This is prohibited for it would model the idiosyncrasies of the training set. Hence, a significant number of cases at each leaf must be retained. Put it another way, the partition must have as few blocks as possible, i.e., the tree should be compact (and comprehensible) and not overfit the data. The forward approach has as an advantage that it is faster in computation time.

In backward pruning, it is argued that the (prune) stopping rules are difficult to get right. Consequently, the tree is grown as far a possible (e.g., when the nodes are still impure) without any regard to overfitting. Then, one looks if some nodes can be pruned away. To be able to do this, a quality for a tree has to be defined in which the size of the tree also plays a role, as it stands for the complexity of the tree. More details follow in section 6.9.2. Hence, a trade-off has to be sought. The terminal nodes are assigned misclassification costs that have to be minimised, taking into account the complexity of the tree. The backward pruning approach is slower in computation time, but a more thorough exploration of possible partitions can be made, resulting in a better class probability tree. The latter makes that it is the most popular approach in C4.5 and CART.

Experience with growing trees in [Breiman et al. 1984] led to the conclusion that for the decrement threshold rule, the number of misclassification were unrealistic low and the tree became too large with regard to the information in the data. The use of more complicated stop rules did not improve this situation: the forward pruning approach failed. Stopping when a threshold was met, may result in a large tree for which the resubstitution estimate of misclassification would approximate zero (overfit). On the other hand, a very small tree will not use some of the classification information available giving rise to an increased misclassification rate (underfit). Therefore, Breiman et al. advocated *pruning instead of stopping*: grow an initial tree that is much too large and then prune it upward (backward pruning).

6.6 Inferencing on terminal nodes: assignment rule?

When class probability trees are used, the distributions in the terminal nodes represent the estimated distribution under the conditions valid for the node (path from root). When one works with decision trees, where preferable a unique class should be assigned to each node, a class assignment rule is trivial. In the more general case of class probability trees, the assignment of which class each (non-uniform) leaf node should correspond to, can be done in different ways.

Depending on the priors (class distribution in training set) and the misclassification costs, a certain class may be picked at the (impure) leaf: this is cost-based classification. The estimated expected misclassification for a case classified as class *i* is given by

$$\sum_{j=1}^{J} C(i|j)p(j|t)$$

A class assignment rule is to select the *i* that minimises the expected misclassification. The resubstitution estimate, $r(t)$, of the expected misclassification for a given node *t* is given by

$$r(t) = \min_{\substack{i \\ i \in C}} \sum_{j=1}^{J} C(i|j)p(j|t)$$

The estimate of the overall misclassification cost for a tree *T* is then determined by the quality of its terminal nodes weighted by the proportion of subjects falling into node *t*, i.e.,

$$R(T) = \sum_{t \in \tilde{T}} R(t) = \sum_{t \in \tilde{T}} r(t)p(t) \qquad (6.12)$$

Under the assumption of equal cost in misclassifying a class *j* object, a simple rule can be used. For a terminal node, one could take as decided class the one for which the conditional probability in that node is maximal, i.e.,

$$\text{decided class} = \underset{j}{\text{argmax}}\, p(j|t) \qquad (6.13)$$

If the frequency approach is used, this becomes the **plurality rule** (use equation (6.4) for estimates)

$$\text{decided class} = \underset{j}{\text{argmax}}\, p(j|t) = \underset{j}{\text{argmax}} \frac{N_j(t)}{N(t)} = \underset{j}{\text{argmax}}\, N_j(t) \qquad (6.14)$$

The estimate of the probability of misclassification $r(t)$ given that a case falls into node *t* is then for a unit cost expressed by

$$r(t) = 1 - \max_j p(j|t) \qquad (6.15)$$

Thus, a simple way to pick the representing class is to take the majority class. Another way is to set thresholds for selecting a class. These thresholds may vary from class to class and one may end up with — when no class exceeds the threshold — an undecided node.

6.7 Advantages and disadvantages of tree classifiers

The advantages are as follows.

+ Classification trees are attractive in that they present a simple and easily understandable structure. The final classification can be cast into rules, which may facilitate acceptance of the model.
+ Classification trees can be applied to any data structure through an appropriate formulation of the questions used for splitting.
+ Classification trees can be used for all kind of predictors.
+ Classification trees use conditional information in handling non-homogeneous relationships.
+ Classification trees do automatic stepwise variable selection and complexity reduction: a search is done at each node for the best split.
+ Classification trees give an estimate of the misclassification probability.
+ In a standard structure, it is invariant under all monotone transformations of individual ordered variables. This is very important for a pattern recognition approach.
+ Classification trees are extremely robust with respect to outliers and misclassification points: one essentially counts cases in a split (frequency based).
+ An important advantage of classification trees over a nearest neighbour method is that the former are based on a procedure for distinguishing between those variables useful for classification and those which are not, [Breiman et al. 1984].

The disadvantages are listed below.

− Classification trees may be too complex, what makes them difficult to comprehend.
− Classification trees may be unstable: if one variable masks another, a small change in (the priors) of the training set may shift a split from one variable to another. Although these splits may almost have the same goodness of split, they can be very dissimilar (see section 6.9.4).
− If one wants to include polythetic splits and Boolean combinations of the variables, the computational burden becomes much higher.
− Classification trees are likely to be more susceptible to unrepresentative data and to overfitting than traditional parametric models. This problem is illustrated in Quinlan [1993], where the author considers an extreme example of a training set of random data in which the class of each case is unrelated to its attribute values. The artificial data set had 10 attributes, each taking a value between 0 and 1 with equal probability. The class was binary with $P(\text{yes}) = ¼$ and $P(\text{no}) = ¾$. A training and test set of each 500 cases was generated. C4.5 build a tree with 119 nodes and had an error rate larger than 35 %. Just choosing ‘no’ all the time would have lead to an error rate of 25 %.
− Greedy algorithms are used to reduce the computational burden. These kinds of algorithms do not guarantee an optimal solution. However, Breiman et al. favour the ‘honesty’ more than the ‘optimality’, [Breiman et al. 1984].

6.8 Machine Learning approaches

ID3 and C4.5 are typical examples from the machine learning domain, where splits are monothetic. Splits are only binary for continuous variables, but they can have any arity for categorical variables. Pruning is forward in ID3 and backward in C4.5. Probabilistic classification is accounted for, and prediction errors are treated heuristically.

6.8.1 ID3: A 'Primitive' Tree Classifier

The ID3 (Inductive Dichotomizer 3) algorithm was devised by Quinlan [1986]. The algorithm is not complex and can process quite large data sets. It uses a splitting rule based on the Shannon entropy. A stopping rule is triggered when the entropy of a leaf node is zero (deterministic determination). Hence, forward threshold-based pruning is used. A 'windowing' process was introduced that enables ID3 to cope with very large data sets by gradually perfecting the tree by looking for instances that are not properly represented and thus by modifying the tree accordingly (on-line learning). The splitting rule in ID3 is based on information gain, which is always positive, see Table 6.2.. The correspondence between the notation of Quinlan and the notation in this thesis is depicted in Table 6.2.

ID3	Impurity measure
S = set of cases[5] in a node	Y is a node
info(S)	$H(Y) = \text{info}(Y)$ (equation (6.9))
$\text{info}_X(S)$	$H(Y \mid X)$ where X is a split for a node Y (equation (6.11))
$\text{gain}(X) = \text{info}(S) - \text{info}_X(S)$	$H(Y;X)$

Table 6.2 : Correspondence between ID3 notation and entropy notation

ID3 may derive decision trees that are absolutely correct, but nevertheless too complicated to understand by humans. Very complicated rules and decision trees are intrinsic to the considered paradigm, see [Gaines 1996].

Deepening of a tree is stopped when the purity measure exceeds a certain threshold. In that case, leaves may no longer contain pure nodes, but instead a frequency distribution over classes. Classification is cost-based or done via the plurality rule.

6.8.2 C 4.5: A 'Classic' Tree Classifier

C4.5 is an extension of ID3 that accounts for missing values, continuous attribute value ranges, (backward) pruning of monothetic decision trees, rule derivation, incremental tree building, and so on [Quinlan 1993]. The splitting rule in C4.5 tries to maximise the information gain ratio. The latter should solve a serious deficiency of the information gain criterion, which has a strong bias in favour of tests with many outcomes. This behaviour becomes obvious if some unique identifier is included in the training set. The gain criterion will select the split where each subnode contains just one case, because $\text{info}_X(S) = 0$. To alleviate this problem, a kind of normalisation can be used. For that purpose, Quinlan defines the term 'split info' by looking at the potential information generated by the child node distribution. The

[5] In CART, S stands for the set of possible splits.

gain ratio is then defined to be the proportion of useful information generated by the split, i.e., the information relevant to classification divided by potential information gain, via

$$\text{gain ratio}(X) = \frac{\text{gain}(X)}{\text{split info}(X)} = \frac{H(Y;X)}{H(X)} \qquad (6.16)$$

Hence, the distribution over the child nodes themselves plays a role via $H(X)$. This gives a kind of complexity penalisation in the splitting.

C4.5 uses backward pruning. An overfitted tree is produced after which irrelevant branches are pruned away by looking if the predicted error rate is better if a branch is replaced by a leaf. Quinlan calls this error-based pruning. The predicted error rate computation is based on internal estimates, which are too optimistic. Therefore, Quinlan decided to do a 'statistical' correction to make the prediction errors less optimistic (pessimistic estimate). Due to the backward pruning, a class distribution is often present in a leaf. The plurality rule is used for class assignment.

C4.5 clearly illustrates the link with the rule-paradigm (and the ML world). It comes with a companion program for generating a set of rules from the decision tree. It goes further than just listing the paths from the root node as rules. It also tries to generalise each rule, and it can create default rules. Even conflict resolution is implemented. All this is aimed at a more comprehensible model (simple rules).

6.9 A Statistical Domain Tree Classifier: CART

In this section, attention is solely devoted to the CART methodology, the motivation of which is statistical prediction. This section is based primarily on the book of [Breiman et al. 1984]. CART is the acronym for 'Classification and Regression Trees'. There is another much less popular statistical tree classifier called CHAID, which is an acronym for 'Chi-squared Automatic Interaction Detection'. However, CHAID only works with categorical variables. It allows higher arity than 2 for splits, but this implies some drawbacks such as a bias toward variables with more splits and the fact that a variable can only be used once for a split. Therefore, CHAID will not be described here.

In the CART methodology, a response variable can be quantitative or nominal while prediction variables can be nominal, ordinal or continuous. A unique backward pruning method is used. It is called minimal cost-complexity pruning. The trees are always binary. The size of a final binary tree is determined by cross-validation and backward pruning. Class assignment in a leaf happens by the plurality rule (or cost based).

6.9.1 Splitting rule

Breiman et al. [1984] conclused that the overall misclassification rate of the tree constructed is not sensitive to the choice of a splitting rule, as long as it is within a reasonable class of rules. A white paper from [Salford 1999], however, claims the contrary. Fortunately, this discussion is of no importance for this thesis, because the Gini rule is always used in class probability trees (see section 5.10.1) or in regression trees (see section 6.10). The Gini rule favours a split into one small, pure node and a large, impure node. The final conclusion from Breiman et al. is that '*Gini splits generally appear to be better*'. Hence, the Gini rule will be taken. The Gini rule assumes unity misclassification costs. An extension of the Gini rule to include misclassification costs can be obtained, but this extension is not necessary in this thesis.

6.9.2 Stopping rule

In CART, an initial tree, $T_{\max}$, is grown that is much too large. Selectively pruning this tree upward, results in a decreasing sequence of sub-trees. Subsequently, cross-validation or test sample estimates are employed to pick out that subtree having the lowest estimated misclassification cost. The criterion to prune or recombine upward is based on misclassification cost.

So, the first step is to grow a very large tree by letting the splitting procedure to continue until all terminal nodes are either very small (default is $N < 5$ or 10) or contain only one class. The initial tree should be large enough and preferable contain pure nodes. The initial tree can be improved somewhat by pruning away descendant nodes t_R and t_L which are such that $R(t) = R(t_L) + R(t_R)$. Continuing this process, results in a new $T_{\max}$, that has the same misclassification cost as the original maximal tree.

In a second step, a nested sequence of subtrees is created by backward pruning of $T_{\max}$. A 'selective' greedy pruning procedure is employed in which each subtree selected is the 'best' subtree in its size range. A complexity measure (or penalty function) is used in this effort. The corresponding process is called (minimal) cost-complexity pruning. A natural way to introduce the required complexity penalty is to look at the number of nodes in the tree, which has the same order as the number of terminal nodes. In minimal cost-complexity pruning, a cost-complexity measure $R_\alpha(T)$ is defined as a linear combination of the misclassification cost of the tree and its complexity penalty (determined by the number of leaves $\left|\tilde{T}\right|$):

$$R_\alpha(T) = R(T) + \alpha\left|\tilde{T}\right| \qquad (6.17)$$

where α is a positive real number, called the complexity parameter (complexity cost per leaf) and where $R(T)$ is determined by equation (6.12). The goal is to find for each α that subtree $T(\alpha)$ that minimises $R_\alpha(T)$, i.e.,

$$R_\alpha(T(\alpha)) = \min_{T \le T_{\max}} R_\alpha(T)$$

The smallest minimising subtree can then uniquely be defined. It can be proven that for every value of α, there exists a smallest minimising subtree. The algorithm has the tendency to remove at first large subtrees with many terminal nodes. The result is a nested sequence of minimal cost-complexity subtrees starting from $T_{\max}$ up to the root node. Each subtree T_k is considered as a classifier on its own.

The final step consists of picking one of these subtrees as optimal. The resubstitution estimate will always yield the largest tree, $T_{\max}$, as optimal. Hence, a better method is to use true estimates to select the right sized tree from among the pruned substructures. These more accurate estimates can be obtained by an independent test sample in case the sample size is large or by cross-validation with smaller sample sizes. As a tree is pruned upward, behaviour like the one in Figure 5.8 is to be expected. Consequently, an optimum must be sought as described in section 5.10.4. The right sized tree is selected by taking the (sub)tree T_k (k is a labelling index) that has the minimum misclassification cost with regard to honest estimates:

$$R^{ts}(T_{\text{candidate selected}}) = \min_k R^{ts}(T_k) \qquad (6.18)$$

Most simulations of $R^{ts}(T)$ as function of the number of terminal nodes reveal a graph with a reasonably rapid initial decrease followed by a long, flat valley, and then a gradual increase as the number of terminal nodes increase. The position of the minimum in this valley is unstable. Therefore, as in section 5.10.4, a 1 SE (one standard error) rule is used to reduce this instabil-

ity and to choose the simplest tree, T_{selected}, whose accuracy is comparable to $\min_k R(T_k)$ within one standard error, i.e.,

$$R(T_{\text{selected}}) \leq R(T_{\text{candidate selected}}) + se(R(T_{\text{candidate selected}}))$$

6.9.3 Class probability trees and oblique trees

The use of class probability trees does not change much to the methods used for classification trees. In class probability trees (see section 5.10.1 for a formalisation of class probability estimators), one does not decide on a class in a terminal node, but one retains a class distribution. Class probability trees look very much like classification trees. Only the Gini index plays a double role: as splitting rule, and as misclassification estimate. For more details, the reader is referred to [Breiman et al. 1984; Van Welden 1998].

The use of oblique trees does not change the methodology either (apart from inserting extra variables for splitting). Accordingly, in CART, oblique trees can be induced. Breiman et al. consider three approaches to splitting

1. use a linear combination of continuous predictors
2. use a combination of Booleans (conjunctive form). It works only for binary variables.
3. add features or predefined combinations

A problem is that their algorithms may be trapped in a local maximum. Another problem is that there are too many terms and hence the split may become uninterpretable. Therefore, a backward and stepwise deletion of terms is proposed by deleting the least important terms first (give smallest decrease) until the effect of the deletion is a certain (predefined) fraction of the most important term (determined before). Hence, it is a subjective approach.

6.9.4 Surrogate splits

In standardised data, missing values can automatically be handled in tree construction and in prediction. The concept of surrogate splits, which is typically for CART, makes this possible. A surrogate split is a split that is strongly associated with the best split: it sends cases to the descendant nodes in the same way as the best split. This association remains, if the surrogate split just reverses the roles of left and right node. Hence, to encompass this situation, complementary splits have to be included.

Besides the best split s^* (see equation (6.2)), consider a split s_m on a variable x_m for the same node t. If s_m is used to predict s^*, then the probability that this happens correctly is given by the probability that both, send cases to the same descendant node. This probability is denoted by , $p(s^*, s_m)$. The error probability is then $1 - p(s^*, s_m)$. The best split s_m, i.e., the one that has the best similar behaviour in sending cases to the descendant nodes is a surrogate split (delivered by attribute x_m), i.e., $p(s^*, \tilde{s}_m) = \max_{s_m} p(s^*, s_m)$. A surrogate split should do at least better than in the naïve case. A measure of association is based on the ratio of the error probability by the surrogate split and the error probability by the naïve split:

$$\frac{\text{error probability by surrogate split}}{\text{error probability by naive split}} = \frac{1 - p(s^*, \tilde{s}_m)}{\min(p_L, p_R)}$$

For a surrogate split to make sense, it must outperform the naïve choice, thus $1 - p(s^*, \tilde{s}_m) < \min(p_L, p_R)$. Consequently, Breiman defines the predictive measure of association between $\tilde{s}_m$ and s^* by

$$\lambda(s^*|\tilde{s}_m) = 1 - \frac{1 - p(s^*, \tilde{s}_m)}{\min(p_L, p_R)}$$

Hence, a good surrogate split must result in $\lambda(s^*|\tilde{s}_m) > 0$.

A surrogate split is searched for each attribute x_m. However, depending on the attribute, some may give a better split than other attributes. The idea is to compute the predictive measure of association for each variable and to order them from best to worst. The algorithm looks as follows:

1. for all variables x_m (main loop)
   find $\tilde{s}_m$ (secondary loop) and determine $\lambda(s^*|\tilde{s}_m)$
2. retain all surrogate splits with $\lambda(s^*|\tilde{s}_m) > 0$
3. sort them in descending order according to $\lambda(s^*|\tilde{s}_m)$

Surrogate splits and missing data

Surrogate splits are not used in the tree construction.

The situation is different for prediction purposes. A case with missing values, such that s^* cannot be used, will be processed by a surrogate split in sending it further down the tree. From all candidate surrogate splits, the best that can process the case, is taken (if the first one cannot, take the second one, …).

This looks much like the case of using several masks in SAPS !

A higher percentage of missing values in the design set may give less accurate trees, and the prediction may be less accurate when missing values come into play.

Surrogate splits and variable ranking

When a best split is chosen, based on impurity reduction, other splits in the same node may give nearly the same impurity reduction. Hence, should the initial variable not be present, the second best would have been chosen. It is said that the second is masked by the first variable. It would be interesting to know how important variables are in splitting and to have a possibility to assign a rank to these variables. For that purpose, two situations can be distinguished:

1. the second best split is possibly not associated much with the best split. For example, it sends other cases to the child nodes, but the reduction in impurity is nearly the same as for the best split. This is called a competitive split. It explains why tree classifiers may be susceptible to relatively small changes in the data.
2. the seconds best split is a surrogate split, i.e., it must have a strong association with the best split and it must give a good impurity reduction too.

A measure of importance should not be based on competitive splits (first case), because they may still have a large impact lower in the tree. Consequently, the importance of these kinds of alternative splits could be too optimistic. This drawback is less likely in the second case, because the second best is like the one that is actually used (its effect is more local and thus less optimistic). Hence, it is in the second case that one defines a measure of importance of a variable x_m by (in case of a association strength tie, take the one with largest reduction of impurity) by

$$M(x_m) = \sum_{t \in T} \Delta i(\tilde{s}_m, t) p(t)$$

Importance is related with ranking, so a kind of normalisation can be done; e.g., set the most important to 100. Furthermore, the more correlation there is between variables the better surrogate splits can be, and this may raise the importance of a variable. It is commented in [Van Welden 1998] that it is a bit strange that the sum over all nodes is taken. A variable can have a low impact in, say all nodes, or it can have a high impact in fewer nodes. The measure of importance defined above will not distinguish between these two situations. A suggestion is given in the reference.

6.10 Regression trees

If the output (response) variable Y is continuous, regression trees can be used. They can be though of as a histogram estimate of a regression surface. Many procedures, such as variable combinations, surrogate splits, missing value handling, and variable importance for tree classifiers just carry over without any modification. Regression trees are simpler because the same impurity criterion is used to grow and prune the tree, and each case has the same weight in 'misclassification'.

An example is given in the book of Breiman et al. [1984]. It is based on a paper from 1978 from Harrison and Rubinfeld. Boston housing values data were gathered to see if there was any effect of air pollution concentration on housing values. Fourteen variables were measured. The response was the median value of homes (in thousands of dollars). Hence, the response is clearly continuous. A regression tree was grown from which a part is shown in Figure 6.6.

A number within each node is the average of the response. The numbers in the middle of the arcs show the cases that go left or right, and the numbers below the leaves are their standard deviations of the corresponding response. The test at each internal node, which is denoted by a circle, is written just below it. The example showed that only 4 of the 13 predictors were retained for splits. Surrogates can be determined to assess the importance of each variable.

In regression trees, the classification rule becomes a real-valued function. Its accuracy is now measured for a large test sample in the least absolute deviation (LAD) case by

$$R^{ts}(d) = \frac{1}{N^{ts}} \sum_{(\vec{x}_n, y_n) \in L_2} |y_n - d(x_n)|$$

or in the least squares (LS) case (mean squared error) by

$$R^{ts}(d) = \frac{1}{N^{ts}} \sum_{(\vec{x}_n, y_n) \in L_2} \left(y_n - d(x_n)\right)^2$$

where in both cases, L_2 is the test set and N^{ts} its number of objects (similar replacements can be done for the cross-validation approach).

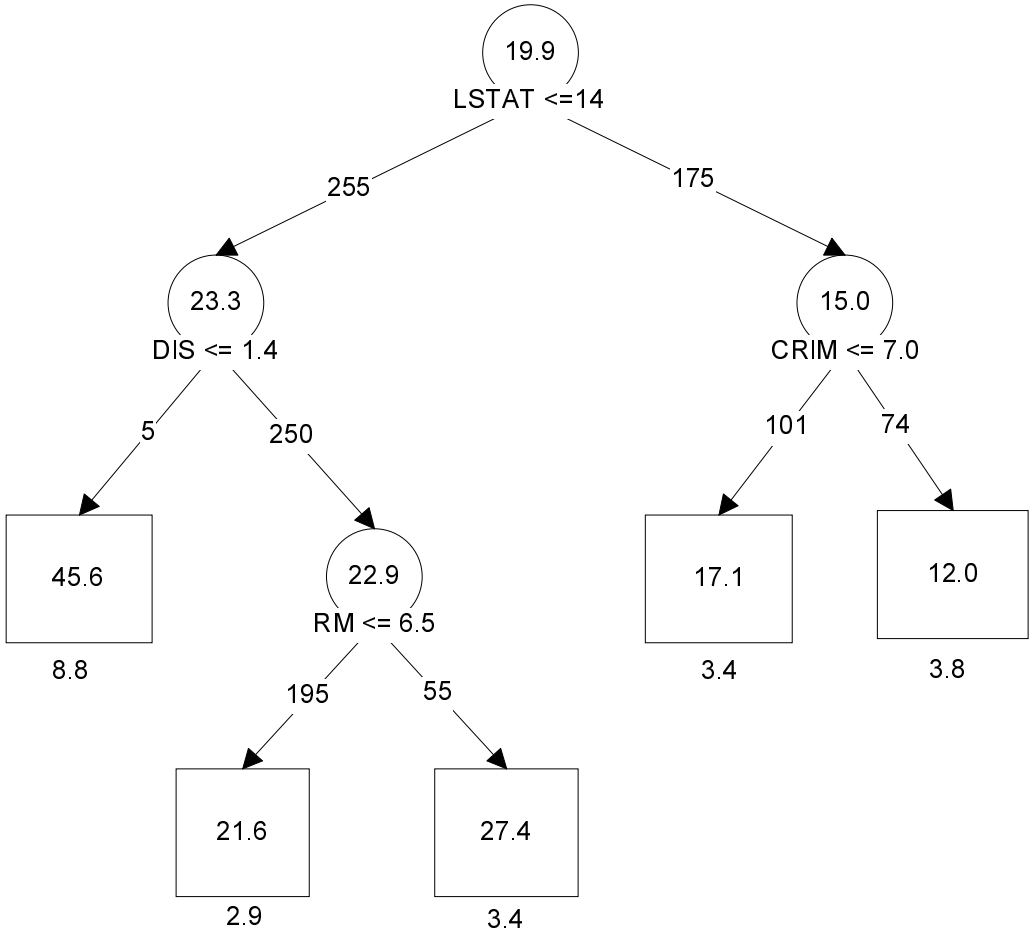


Figure 6.6 : Regression tree example (part from figure in [Breiman et al. 1984 p 219])

While a misclassification rate has an intuitive interpretation, the mean squared error has not. To remove scale dependency, an estimated relative mean squared error, $RE^{ts}(d)$, is defined by dividing the mean squared error by the sample variance, i.e., $RE^{ts}(d) = R^{ts}(d) \big/ R^{ts}(\bar{y})$, where $\bar{y}$ is the sample mean.

The splitting rule is now based on minimising the resubstitution estimate of misclassification. Hence, the best split is the one that minimises the weighted variance (over the child nodes) in the LS regression and that minimises the sum of the absolute deviations from the node medians in the LAD regression. More details can be found in [Breiman et al. 1984].

Backward pruning is based minimal error cost-complexity, which is very similar to the tree classifier case, but with the misclassification rate simply replaced by the error measure. It is typical that the valley containing the minimum value of the misclassification error is flatter and wider than in classification trees. As for tree classifiers in CART the 1 SE rule (one standard error rule) is used for selecting the best tree.

Regression trees are reported to be competitive with linear regression. They can be more accurate on non-linear problems, but less on linear problems (which is consistent with the general principle of using parametric models when the parametric assumptions hold).

6.11 Conclusion

The general principles of tree classifiers were explained. From the two basic pruning approaches, forward and backward pruning, the latter is the most popular. An optimal tree cannot be found, hence, (greedy) heuristics are necessary to find a sub-optimal tree. The same methodological optimality problems are present in SAPS, see chapter 4.

Breiman et al. [1984] state that *within a wide range of splitting criteria the properties of the final tree selected are surprisingly insensitive to the choice of splitting rule*. Murthy [1997] shows that the use of more extensive search heuristics than the traditional greedy heuristic is unnecessary, and often harmful. Therefore, a restriction to the two most popular tree classifiers, C4.5 and CART, which use greedy heuristic evaluation functions combined with backward pruning, is a representative and meaningful choice to illustrate a data mining approach to SAPS. However, CART has the additional advantage that it provides regression trees as well as oblique trees, that it penalises complexity, and that it gives a variable ranking. In addition, CART's theoretical fundaments are more statistically sound.

Research is still very active in classification and regression trees, see [Murthy 1997; Shang and Breiman 1996; Blockeel 1998] for example. Including all this new research in this thesis is virtually impossible and it would obscure the main line followed (since this thesis is not about how to improve the performance of tree classifiers). In that sense, one could say that CART is the 'worst' performer when comparing tree classifiers with SAPS. The reader should bear this in mind when reading chapter 8 and the appendices.

References

Armitage P., Colton T. [1997], Encyclopedia of Biostatistics. Wiley, 1997.

Berry M. J. A., Linoff G. [1997], Data Mining Techniques For Marketing, Sales and Customer Support. Wiley Computer Publishing, 1997.

Blockeel H. [1998], Top-Down Induction of First Order Logical Decision Trees, Ph.D. thesis (in English), Katholieke Universiteit Leuven, Leuven, Belgie, 1998.

Breiman L., Friedman J. H., Olshen R. A., Stone C. J. [1984], Classification and Regression Trees. Chapman & Hall, 1984.

Gaines B. R. [1996], "*Transforming Rules and Trees into Comprehensible Knowledge Structures*" Advances in Knowledge Discovery and Data Mining, ed. Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R., AAAI Press/MIT Press, Cambridge, England, p. 205 - 226, 1996.

Hyafil, Rivest [1976], cited in [Quinlan 1993].

Kodratoff Y. [1997], "*From the Art of KDD to the Science of KDD*", Learning, Networks and Statistics, CISM Courses and Lectures no 382, ed. Riccia G. D., Lenz H-J., Kruse R., Springer, Wien, p. 135-160, 1997.

Mingers J. [1989], *"An Empirical Comparison of Selection Measures for Decision-Tree Induction"*, Machine Learning, 3(4), 319-342, 1989.

Murthy K. V. Sreerama [1997], On growing better decision trees from data. Ph.D. thesis, Johns Hopkins University, Baltimore, Maryland, 1997.

Quinlan J.R. [1986], "*Induction of Decision Trees*", Machine Learning, vol. 1, p 81 - 106, 1986

Quinlan J.R. [1993], C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning, 1993.

Rulequest [1999], see www.rulequest.com

Salford [1999], see www.salford-systems.com

Shang, N., Breiman L. [1996], "*Distribution Based Trees Are More Accurate*", Proceedings of the International Conference on Neural Information Processing, Hong Kong, p. 133-138, 1996.

[Shapiro 1987] cited in [Quinlan 1993].

Van Welden D. [1998], Tree Classifiers as Data Mining Tools. MSc. Thesis. Catholic University of Leuven, Belgium, 1998.

Chapter 7

Relationship between General System Theory and Knowledge Discovery in Databases via Meta-Modelling

7.1 Introduction

General System Theory and Knowledge Discovery in Databases have a lot in common. In this chapter, this commonality is described from a high abstract level to a more concrete level. At an intermediate level, a comparison between GSPS and KDD is done. Further focus will be given to the used paradigm in GSPS (already described in chapter 3) and the supervised learning paradigm (described in chapter 5).

The comparison starts by considering the philosophical aspects of GST and KDD. Next, their life cycle is compared and emphasis shifts are explained. At the next level of comparison, it is shown how in GSPS, a time-invariant pattern or mask 'flattens' input-output data into the state-observation space (chapter 3), such that the state-observation records are static. Still, many masks could be tried for the mapping. Fortunately, taking the idea used in the sub-optimal mask search, which starts the search from a maximum allowable complex mask, a step further, a stream of data records that leans itself perfectly for a KDD approach can be obtained. As mentioned in part one of this thesis, the line of research that is being pursuit, is one of system identification of directed systems. It corresponds to the supervised learning paradigm in KDD where outputs of a system under investigation form the responses known at each time instance. Consequently, different data-mining methods can now be applied to find patterns in the state-observation matrix. They may constitute a new powerful approach to the identification of dynamical systems, which extends the possibilities of SAPS in dealing with different type of variables, in handling larger databases, in effectively coping with missing values, and so on.

7.2 Comparing General System Theory and Knowledge Discovery in Databases

7.2.1 Abstraction of a model

In the sequel, a model is defined as any formalisation that aids in understanding a system under investigation. This definition is very abstract, which makes it applicable to the GST and the KDD domain. Hence, a model in its most general form can be (the list is not exhaustive):

1. a set of differential (or difference) equations, a block diagram, a Petri net, a Bond graph, any qualitative model (e.g., confluences), a time series model, …

2. a classification or regression tree, a dependency network, a hypothesis, a regression model (linear, non-linear, logistic, …), a discrimant function, …
3. a set of rules (if-then rules, association rules, …), a Markov chain, a look-up table or contingency table, a set of data (for lookup), a clustering model, a neural net, a genetic algorithm, …

The first types of models are mainly known in the domain of GST, the second type in the domain of KDD, and the third types are commonly used in both domains. The above definition of a model is consistent with the definition of Minsky [1965] that states: "*An object 'A' is a model of an object 'B' for an observer, if the observer can use 'A' to answer questions that interest him about 'B'* ".

7.2.2 Philosophical issues

In GST, system observations may be obtained either actively or passively. In KDD, the latter situation is the rule (although ML considers both cases, see chapter 5), while in GST both settings deserve equal attention.

The top-down approach to modelling is a deductive approach. It can be compared to hypothesis testing in KDD. In both cases, one has a priori knowledge that has to be cast in a formalised form (a model in its most general meaning). A difference in the deductive approach is that general systems theory has a long tradition to handle systems and sub-systems (decomposition of systems and coupling of sub-systems), while KDD usually tackles one (indivisible) system.

The bottom-up approach to modelling is better known in general systems theory as system identification (chapter 1). Klir calls the bottom-up approach 'investigative', [Elzas 1984]. The latter term adheres more to terminology used in KDD, where such an approach is known as data exploration (chapter 5). Hence, an inductive approach starts from the data and tries to induce a model (in its most general meaning). Both approaches are depicted in Figure 7.1.

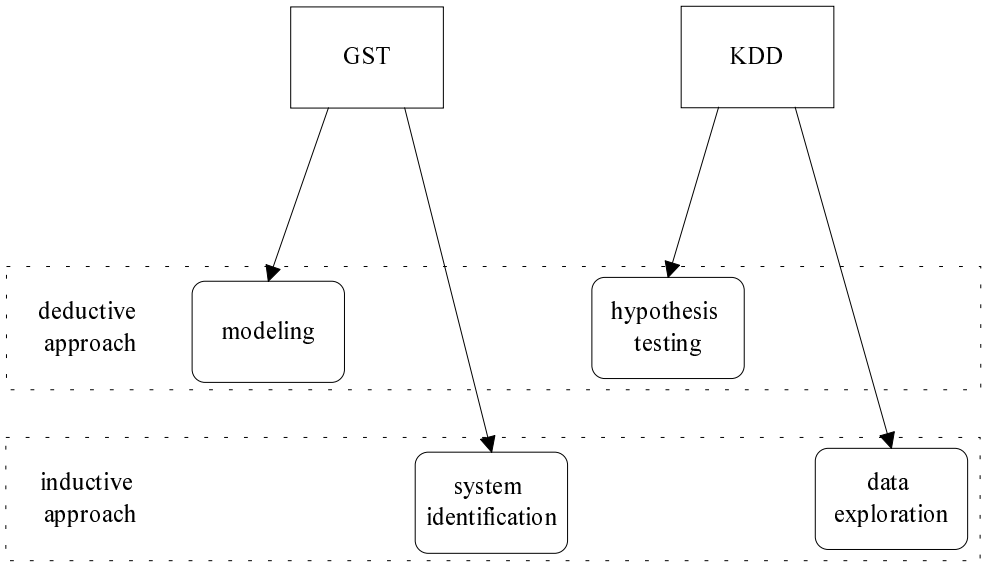


Figure 7.1 : Deductive and inductive approach in GST and KDD

In general, KDD emphasises more the inductive approach, which makes it better suited for 'black box' and 'dark-grey' systems. Consequently, its success is mainly found in the corresponding domains of science, where little or no a priori knowledge is available. By using the rainbow of Karplus in Figure 7.2, one notices that most applications of KDD lie in the field of

economics, sociology, ecology and physiology. Additionally, Figure 7.2 shows that this restriction is not crisp.

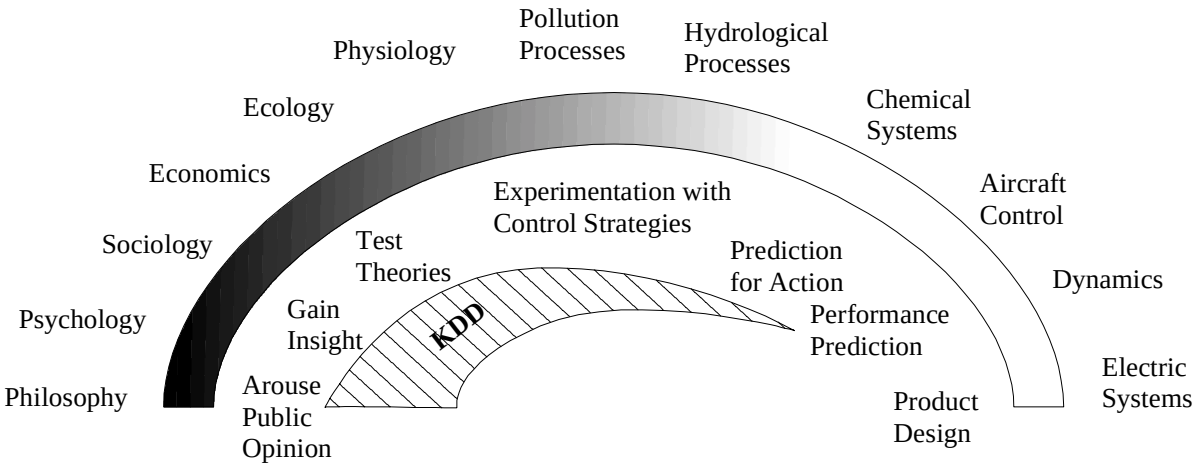


Figure 7.2 : Spectrum of modelling

With regard to the black box aspect, it is interesting to look at the definition of a system given by Klir in chapter 1. It allows for both the deductive or inductive approach. From this definition, it can be seen that a class of systems can be uniquely defined by given activity (series of I/O records). This corresponds with a system identification approach in GST and with the starting point of data exploration in KDD. The first three traits are commonly used in KDD, while the latter two, UC and ST structure, are more devoted to GST.

7.2.3 The life cycle of GST and KDD

Klir [1985] considers meta-data in GSPS, but meta-data has more semantic richness in KDD. A first aspect of this richness can be illustrated by looking at the data types in a data warehouse, which are shown in Figure 7.3.

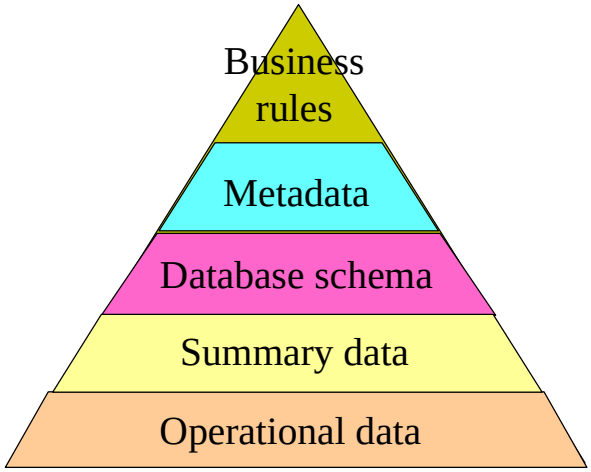


Figure 7.3 : The use of meta data in data warehousing (from [Han 1999])

Operational data is the data itself. In data warehousing, this also means where it comes from, when it was stored, etc. Summary data gives summaries of the data, so it is a kind of meta-data already. The database scheme gives the physical layout of the data. The meta-data level itself is the logical model. The meta-data repository also encompasses business terms and definitions, ownership of data, and charging policies. For example, operational meta-data concerns:

- data lineage: history of migrated data and sequence of transformations applied
- currency of data: active, archived, purged
- monitoring information: warehouse usage statistics, error reports, audit trails

A second aspect of the richness of the term meta-data in KDD, is found in the data mining approach. Here, meta-data says something about the measurement scale; whether variables are derived from others (data projection); if there are bounds on data; if there are structural zero's; whether there are distributional assumptions made (parametric versus non-parametric), etc. Meta-data is thus much related to the use of a-priori knowledge in data-pre-processing (structural zero's, derived variables), and in model specification and selection (which data mining method to use). Remark that the distinction between data and meta-data may be blurred.

The quality of the data structure on the database determines to a large extend the success of the application on the functional level as well as on the management level. A general framework that serves this purpose, and which is a synergy of the life cycle of 'classical' modelling (see Figure 1.5) and KDD (see Figure 5.1), is depicted in Figure 7.4. It shows the similarity between general systems theory and KDD in a broad context.

Goal formulation

All problem statements start with the identification of the problem. They try to set a goal in order to solve the problem at hand. To be able to meet a goal, it has to be formalised so that a rigorous approach to problem solving can be obtained. For that purpose, a focus on part of reality has to occur, i.e., one has to define a system. Defining a system is studied in the general systems theory by Karplus [1976], by Klir [1969], and by Zeigler [1976] (list is not exhaustive). It involves defining the boundaries of the system, the interaction of the system with the environment, and last, but not least, defining the relevant variables (space-time resolution). These are concepts that are described in GST, and which are as well applicable to KDD because general system theory is applicable for any model (thus also to data mining).

With regard to goal formulation, more similarities are present: e.g., gaining insight (in KDD) corresponds with understanding (in Systems Theory). Both domains stress the principle that a model should not be more complicated than absolutely necessary (Occam's razor). The three distinguished broad levels of intervention in GST, i.e., management, control and design, are equally applicable to the domain of KDD. Obviously, data mining can be used for gaining insight (model purpose), and the pattern found in KDD provides an opportunity to intervene at some level (as in modelling). However, GST puts some more emphasis on a correct classification and prediction. To have a speedy or less costly classification and prediction is less the issue in GST (except perhaps in the domain of control theory), but of more importance in KDD. This can be viewed implicitly in the interestingness function defined in chapter 5. Hence, the 'Quick decision' problem type, where accuracy may be less important than comprehensibility, is less stressed in classical modelling[1]. A similar argument applies for the cost of a model.

With regard to the goal formulation, both domains have a lot in common. Hence, it is better to speak of emphasis shifts in the goal type that is considered most prominent.

[1] Although it has been considered very briefly in [Elzas 1984].

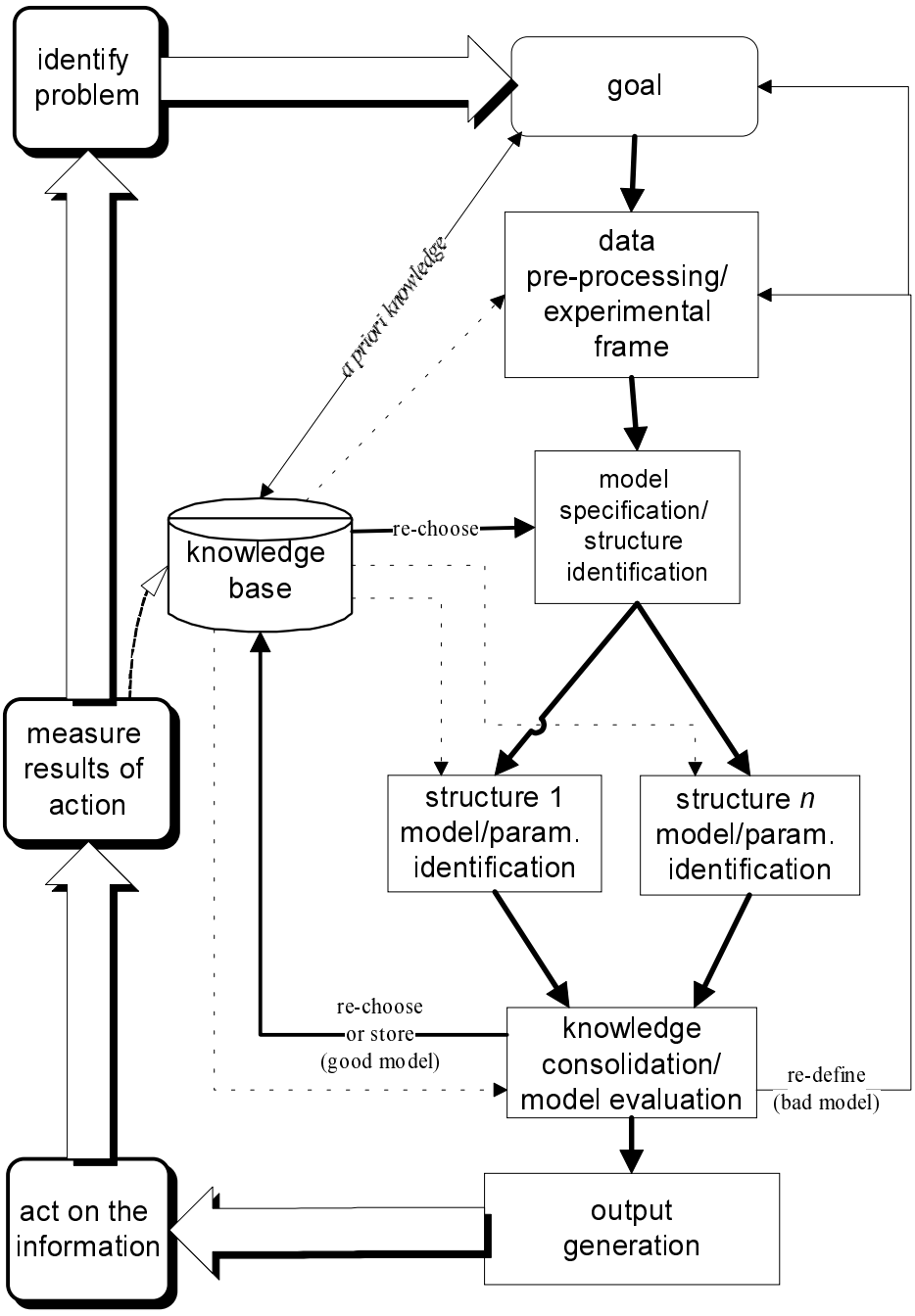


Figure 7.4 : General modelling and KDD

Data pre-processing versus experimental frame definition

Defining a system has as much to do with the definition of the object system (as defined by Klir), as with an experimental frame (as defined by Zeigler). An experimental frame isolates specific input/output behaviour both of the real system and its model, [Elzas 1984]. In KDD, this corresponds with the data pre-processing step. Attribute focusing is in fact nothing more than defining what are supposed to be relevant variables for the system under investigation. This is consistent with the notion of an experimental frame and with what Klir calls an observation channel. It also fits in the viewpoint, taken in systematic modelling, which states that a model is conceived as a collection of variables and relations among them, [Ören 1984]. The collection of relevant variables is attribute focusing, while identifying the relations is part of

the data-mining step. The KDD approach to attribute focusing is more related to the philosophical viewpoint of Klir than that of Zeigler. This can be induced from what Elzas [1984] wrote: "*Imagine that we are able to prove that the model is not sufficiently detailed in its system description for some goal that was set beforehand. Confronted with this situation Klir will remark that the reason for this deficiency is a lack of knowledge, while Zeigler will consider the observation-frame to be insufficiently detailed*"[2]. Data focusing is less used in systems theory, because the nature of data is often different than for KDD. In KDD, one usually starts with a collection of static independent records. Taking a relevant subset poses no major problems. However, in general systems theory, taking a subset of time-depending data is usually not done: data records do depend on each other. This does not mean that GST has no way of dealing with an abundance of data records. A kind of data reduction can be done in GST via the determination of the Nyquist frequency and via filtering techniques. Still, it illustrates the static(KDD)/dynamic(GST) modelling aspects of the respective domains. On the contrary, data projection can be applied for both domains. It can be meaningful to create a new variable, which summarises the behaviour of a set of other variables, and to use that new variable in the modelling phase that follows later (data projection also fits in the definition of an experimental frame). Finally, data cleaning is appropriate whenever values are missing. Missing values are not really considered in systems theory. Nevertheless, they may appear in real systems. This is especially true when considering system identification. For many problems in systems theory (think about controllers, and other man-made devices), there are no missing values by design. Hence, modellers in the domain of systems theory do rarely meet systems for which there are missing values.

Model specification versus structure identification

The data-mining step in KDD is much related to modelling in system theory. Model specification can be compared with model structure identification. It involves deciding what type of model to use (in system theory: Bond graphs, Petri nets, block diagrams, … in data mining: classification trees, hierarchical clustering, linear regression, neural networks, etc.). The model specification step is straightforward applicable to both domains and can be considered at different abstraction levels. On a very abstract level this involves choosing if one wants to use Bond graphs, Petri nets, block diagram, rules, neural nets, etc. (for systems theory) or trees, clustering, rules, neural nets, regression models, etc. (for KDD). On a more concrete level, one has to further specify the model. Examples are: linear, logistic, or non-linear regression (KDD), linear or non-linear models (GST), state-space or transfer functions in block diagrams (GST), kind of neural net (both domains), kind of tree (decision, regression, …) (KDD), order of a differential equation (GST), type of clustering (KDD), etc. In both domains, the goal has a large impact on the used model types: when comprehensibility is more important certain representations may be more preferred than others. For example in GST, a block diagram may be more comprehensible than a Bond graph (it is also relative to the field of expertise). A tree structure is more comprehensible than a neural network (KDD). Rules are more comprehensible than some other model types (both domains), and neural networks are usually the least comprehensible (both domains). The issue about comprehensibility has a lot to do with the greyness of the model: black box models are always less comprehensible than white box models (if complexity issues of a particular model type are ignored).

The amalgamation of terminology from GST and KDD may shed new light on the terms 'model specification', 'model structure', and 'model complexity'. Model specification has a lot to do with the goal setting, while the model structure is more determined by the experi-

[2] Some text between brackets or '--' in Elzas [1984] has been deleted.

mental frame. Model complexity is related to validation and parameter estimation. This is not to be confused with model evaluation (evaluating model specifications).

An example of a potential (not complete) taxonomy is found in Table 7.1. It shows the terms in relation to each other and it gives an idea of the degree of abstraction that comes into play. Finishing Table 7.1 requires the experience of many domain experts[3], especially if one bears in mind that Table 7.1 only shows one dimension of a larger taxonomy from which Table 7.2 picks another dimension.

Model specification	Rules	Differential equations	Tree classifiers	Neural networks	Time series	…
Model structure	Conjunctive, disjunctive, …	Linear, non-linear, time-invariant, …	Classification, regression, survival …	Kohonen, backpropa-gation, …	AR, ARMAX, MA, …	
Model complexity	Rule size	Order	Number of nodes	Number of neurones	Order	

Table 7.1 : Model specification, structure and complexity

Another aspect is that the KDD society is used to handle all kind of variables (nominal, ordinal, continuous, etc). The GST society also knows about this taxonomy of variables[4], but they do not have such a systematic approach to constructing appropriate models for them. In the KDD society (especially statistics), one uses different models for nominal variables, ordinal variables and continuous variables (e.g., linear regression versus logistic regression, chi-squared based models on contingency tables versus other). Furthermore, the size of the data set is used as a guideline too. For small data sets, exact methods are used, while for larger data sets, asymptotic methods can be used.

With respect to the model structure, principles of good architecture [Klir 1985], such as consistency, parsimony, transparency, generality, and completeness,… are equally applicable in KDD. This is because a model is an abstract concept, which is so general that it encompasses the patterns that are sought, or the hypotheses tested in a KDD environment. Overfitting issues are very important in the context of model complexity (chapter 8).

Data mining or parameter estimation

Model fitting involves parameter estimation (identification) in modelling. It is also called model calibration, [Elzas 1984]. This reduces to estimating parameters or coefficients in differential equations (GST), parameters in state-space models (GST), regression models (KDD), etc.

Model validation consists of comparing the behavioural data of the system under investigation and the calibrated (fitted) model. Usually, a train-and test method is used. A rule of thumb is that 2/3 is used for fitting the model (training) and 1/3 for validation (testing). Cross-validation is commonly used in KDD, while it is not so popular in GST. Even more specifically, bootstrapping is known too in KDD, but almost unknown in GST. Replicatively validity is known in GST: it consists of fitting the model on the training set. In KDD, this is better

[3] This is not a major concern of this thesis, but it may give an impetus for completion.

[4] The methodological types of Klir, which are described in chapter 1, do a similar thing

known as internal estimates (resubstitution estimates, [Breiman 1984]). Predictively validity is done on a test set; it gives true estimates. What Elzas calls 'realism', i.e., looking at structural isomorphisms at different degrees of lumping of sub-systems (and corresponding sub-models), is not considered in KDD. There is usually only one global level, and the validation takes place for the total (indivisible) system.

Knowledge consolidation and model evaluation

KDD uses an interesting function for *evaluating* a model. In this respect, the more general concept of an interesting function is used. The evaluation can be based on more than just accuracy performance. For example, KDD can take into account what is economically interesting (cost of model) or it can take the faster model with regard to prediction. This may prove an important point for the modelling society when they want to evaluate their models in an economic context (cost of modelling), or when speed is of the utmost importance. KDD provides a more general framework for dealing with these situations.

Model refinement is equally applied in GST and KDD; when a model does not validate well, another model (structure) is chosen. When this fails too, one can go one step further back and redefine the experimental frame or even adjust the goal. The refinement of an existing model is a major issue in modelling. The systems theory as developed by Zeigler (use of a SES (System Entity Structure) and model base) has as purpose to construct and refine models, [Zeigler 1976], [Van Welden and Vansteenkiste 1992].

From Figure 7.4, it can be seen that many models may be used in parallel. The models can be of a different specification (e.g., neural nets and genetic algorithms), and they can be situated on different epistemological levels (e.g., rules versus decomposed models). In the latter case, one speaks of shallow or deep models (in GST). The concept of shallow and deep models is also found in the KDD environment: report generation is shallower than OLAP, and OLAP is shallower than KDD. In KDD, much attention is paid to comprehensibility of models, so a rule model can be investigated in juxtaposition to a tree classifier. They do not differ much in their expressiveness (modelling power), but the machine learning society does make a difference with regard to how easy they can be understood. Hence, the GST and the KDD community focus a bit different on the models (decomposed models, comprehensibility issues with regard to the model representation).

shallow			**deep**		
Report generation	OLAP	Neural nets, Genetic Algorithms, and lazy methods	rules	Tree classifiers and regression trees	decomposed models

Table 7.2 : Shallow models versus deep models

Models are not only evaluated with regard to an interesting function, or validated with regard to a goal setting, but in KDD, model *specifications/paradigms* are also compared with each other. This belongs to the knowledge consolidation step. Here, model evaluation is on another epistemological level than in the data-mining step. Models are not only compared on a test set to validate the parameter estimation, but they are compared on yet another (independent) test (or evaluation) set to evaluate the chosen model specification. The GST community has not done so much work in this aspect as the KDD community, which focused and implemented this right away (see SAS-enterprise miner, [SAS 1999]). Model switching is used in GST, but there, it is more the purpose to see if a model is still valid for time-dependent behaviour (sequential model replacement). One may remark that SAPS can use different masks in parallel

to improve on the forecasting, but the models are of the same type. Hence, the underlying paradigms are different. They can be considered orthogonal (one deals with comparing models for a same behaviour, the other deals with models for changing behaviour), but they can be used both when studying time-variant behaviour. Consequently, the term 'model switching' is somewhat vague. One can switch from a state-space model to a neural net when the behaviour goes outside the specifications of the state-space model (i.e., when another structure is more appropriate). It could even be that, when the system goes out of the pre-set specifications (and thus the specific model fails), the goal settings change accordingly (e.g., speed becomes more important than accuracy, or robustness is more important than speed). In GST, it is known that under certain circumstances, for example, linear models may have to be replaced by non-linear models or that PID controllers are replaced by fuzzy controllers. However, this is more because of the limitations of the former with regard to a certain experimental frame.

In both GST and KDD, the consolidation with regard to storing the found knowledge is present. In GST, the newly found model is stored in a model base (called modelling in the large, see also [Van Welden et al. 1991]), while in KDD this is not stated so explicitly. Output generation has a lot to do with appropriate graphical representations of the data. In that aspect GST can learn from KDD, because the latter is well acquainted with all kinds of graphs (e.g., multidimensional graphs such as Trellis graphs). Multi-dimensional visualisation techniques are considered very important in KDD (e.g., in statistics, see [Friendly 1991]).

Hence, there is room for elaboration from both sides.

Remarks

- The 'management' part of Figure 7.4 (acting on the information, measuring the results of the action and storing them, perhaps recognise another problem type) is very general. Thus, it is valid for any modelling attempt, be it via GST or via KDD.
- In Figure 7.4, the knowledge base is not split up in a model base and a database of records because a matrix of records can be considered as a model too. Rules, which may serve as a priori knowledge, or even written information in plain English, can be considered as models too, be it of a less formalised level (for the latter). Therefore, a general term could be 'model base', but the option is made to make it even more general by using the term 'knowledge base'.
- When many models compete in the model evaluation step, one has to decide what to do with them: store/use them all in parallel or take only one. Model evaluation does not exclude the use of more models in parallel.
- Both GST and KDD acknowledge the necessity of feedback from steps that appear later in the cycle to steps that appear earlier. This is indicated in Figure 7.4. Therefore, the steps in the life cycle are more intertwined than one should expect at first sight.

7.2.4 Emphasis shifts between GST and KDD

The emphasis shifts are already described in the previous section. Summarised, one can say that

- GST focuses more on accuracy of a model, sometimes speed and less the cost of a model. It does not lay emphasis on data reduction as KDD does. Modelling of dynamical systems is the primary focus of research. Decomposition of models is common place.
- KDD puts more emphasis on static systems (temporal databases can be used and sequential patterns detected, but this is not the same as modelling dynamical systems). KDD fo-

cuses more on model comparison in the large (via a third test set), on data warehousing and relies on a richer semantic meta-data structure. In KDD, one has more experience with very large databases and with high dimensionality problems. Data reduction is commonly used. The interestingness function is more general than the evaluation functions used in GST.

7.3 Integrating GSPS and Data Mining

This section concentrates on the data mining process itself. The data mining approach that is proposed in this thesis goes further than the hill-climbing approach for mask searching. It uses the idea of 'data-flattening' from GSPS and the idea of a maximum allowable mask of SAPS-ST with its use of a (tree) structure. The maximal mask still sets an upper bound on the pattern search space, but now different structures (with different inductive biases) can be used. The step further, thus sits in the plethora of new techniques for pattern identification, which stems from the domain of data mining. When discussing regression trees in chapter 8, a lot more analogies with the hill-climbing approach will spring up.

7.3.1 Converting trajectories to static data in GSPS

Classification, which is discussed in chapter 5, typically works with static records. Hence, to be able to use classification on something like system identification, one has to be able to convert the dynamics in the activity matrix somehow to static data. This can be realised by a mask, because a mask is a time-invariant pattern that typically 'flattens' the dynamical relationship into a static one. For example, consider a candidate output-input dependency with depth 2 in time, i.e., the dependencies between output and inputs (or previous outputs) do not go further than 2 time instances back in time (mask memory depth is 2):

$$y(i) = \tilde{f}(u_1(i-2), u_1(i), u_2(i-1), u_3(i-1))$$

with $\tilde{f}$ a qualitative function, which is represented by a state-observation matrix that is generated by applying the pattern. The mask for the previous equation above is given by Table 7.3.

	inputs				*output*
time	u_1	u_2	u_3	u_4	y
i-2	-1	0	0	0	0
i-1	0	-1	-1	0	0
i	-1	0	0	0	1

Table 7.3 : A proposed time-invariant pattern or mask

Applying this pattern by sliding it over the data gives a static (time-invariant) relationship in the state space, which has the general form as shown in Table 7.4. The index stands for the observation number, or reference time instance that is used in the mask sliding process.

index	state				output
	x_1	x_2	x_3	x_4	y
1	$x_1(1)$	$x_2(1)$	$x_3(1)$	$x_4(1)$	$y(1)$
2	$x_1(2)$	$x_2(2)$	$x_3(2)$	$x_4(2)$	$y(2)$
	…				

Table 7.4 : 'Static' matrix

The state vector at the reference time *i* is given by (see equation (1.9)).

$$(x_1(i), x_2(i), x_3(i), x_4(i)) = (u_1(i-2), u_1(i), u_2(i-1), u_3(i-1))$$

and the new relationship is now expressed as

$$y(i) = \tilde{f}(x_1(i), x_2(i), x_3(i), x_4(i))$$

As a matter of fact, the *i* index can be dropped altogether, thus obtaining simply

$$y = \tilde{f}(x_1, x_2, x_3, x_4)$$

The raw recoded data matrix in Table 7.5 gives a concrete example.

i	u_1	u_2	u_3	u_4	y
0	low	medium	high	low	High
1	high	low	medium	high	low
2	low	low	low	high	low
3	medium	low	medium	low	medium
4	high	high	high	low	very low
5	low	medium	low	high	very high
6	medium	medium	low	low	medium
7	medium	high	high	high	low
8	low	high	high	low	very high
9	high	low	medium	low	medium
10	medium	low	high	high	low
…	…	…	…	…	…

Table 7.5 : Recoded data matrix

With the mask from Table 7.3, the data matrix in Table 7.5 flattens to the static matrix given by Table 7.6.

x_1	x_2	x_3	x_4	y
	low			high
	high	medium	high	low
low	low	low	medium	low
high	medium	low	low	medium
low	high	low	medium	very low
medium	low	high	high	very high
high	medium	medium	low	medium
low	medium	medium	low	low
medium	low	high	high	very high
medium	high	high	high	medium
…	…	…	…	…

Table 7.6 : Example of a static matrix

All sub-masks from Table 7.3 can be found back in the static matrix in Table 7.6 via the deletion of certain columns. Consequently, Table 7.3 acts as a primary mask. To enlarge the search space, it is better to work with a maximal allowable mask to generate a static data matrix with a maximal number of generating variables (most complex model that gives overfit). In addition, the output of a record in the static data matrix is assumed to depend only on the corresponding generating variables, which are now considered as inputs. Hence, the outputs are regarded as conditionally independent (cf. chapter 3) of each other. The newly obtained data matrix will normally be very large with regard to its number of (new) variables (see chapter 8 for realistic examples). In fact, one has achieved some form of data-augmentation. This is exactly what a data mining method is designed for, see the statement of Berry in section 5.4.3. Hence, finding a pattern in this matrix seems a job for a data- mining method.

7.3.2 Working on static data in data mining

Classification happens by simplifying a matrix of static records by searching only relevant (sampling) variables, which can be identified with sub-masks of the original proposed maximum allowable mask. A static matrix, such as the one in Table 7.4, serves as a, usually wide, starting table of static records that is given to the classification algorithm.

Making an attribute (generating state) irrelevant for the classification task implies ignoring the corresponding column. Hence, all time-invariant patterns less complex than the one corresponding with the maximum allowable mask can be found, but ones that are more complex never will[5]. This may seem a severe restriction, but it is not as severe as in the hill-climbing approach of SAPS-ST because the maximum allowable mask can now be taken much deeper. This is thanks to the ability of data mining methods to handle many attributes. Of course, the deeper the maximal allowable mask, the more new variables emerge for the data mining method to process. It is quite easy to determine how many new variables come into play via this method.

[5] It is shown in appendix D how this can be dealt with.

For m variables (output included), a maximal mask of depth d generates ($m \times d$) extra columns besides the original m columns of raw data. This leads to a total of ($m \times (d + 1)$) columns or new attributes, i.e.,

$$(u_1, \cdots, u_{m-1}, y) \xrightarrow{\text{flattening}} (x_1, x_2, \cdots, x_{m(d+1)-2}, x_{m(d+1)-1}, x_{m(d+1)}).$$

The number of new (state) variables is computed in Table 7.7. Consequently, one may end up with a lot of variables, which may even become too many in number, to be processed by a data mining algorithm. In the conclusion of this thesis, a way to cope with this will be mentioned.

		memory depth (d)				
m	inputs	2	7	12	52	365
4	3	12	32	52	212	1464
7	6	21	56	91	371	2562
11	10	33	88	143	583	4026
21	20	63	168	273	1113	7686
51	50	153	408	663	2703	18666

Table 7.7 : Number of variables for a given number of inputs and a given memory depth (one output)

Assuming n records for each column, the total number of data entries is given by $n \times (m \times (d+1)) - d$. Knowing this number is important for data-mining methods that put a maximum on this number (see chapter 8). Of course, there is also a lower limit on the number of data records. It should exceed the maximal mask depth by a multiple of at least five (see chapter 2), but usually this will always be the case for more complex real-world systems. Too low a number of data records leads to a restriction on the maximum allowable mask depth.

7.3.3 Analogies between KDD and GSPS

The life cycle from Figure 7.4 can be made more concrete for GSPS. In the sequel, focus will lie on the KDD approach as shown in the KDD block of Figure 5.1. Figure 7.5 shows the relationship with the epistemological levels of GSPS.

The goal setting and formalisation are situated in the source system, as is the knowledge consolidation (and output generation). The latter two are consistent with the general picture of GSPS (see Figure 1.10) that describes the interface between the general methodology and the domain dependent knowledge.

Attribute focusing can be done a priori and independent of the data values. Therefore, it should be put in the source system. Data cleaning is situated in the data system, because it relies on the data themselves. For dimensionality reduction, one should set or determine derived variables beforehand, so it *should* be put in the source system. This is consistent with the viewpoint of Kodratoff [1997]. In data focusing, one knows what kind of records to look for (source system), but one needs the actual data records to effectively pick out the relevant data records (data system). Consequently, it belongs to both source and data system. This shows some relation with the recoding issues in SAPS: fixed recoding belongs to the source system, but uniform recoding in the data system? Apparently, the GSPS framework has not accounted for this.

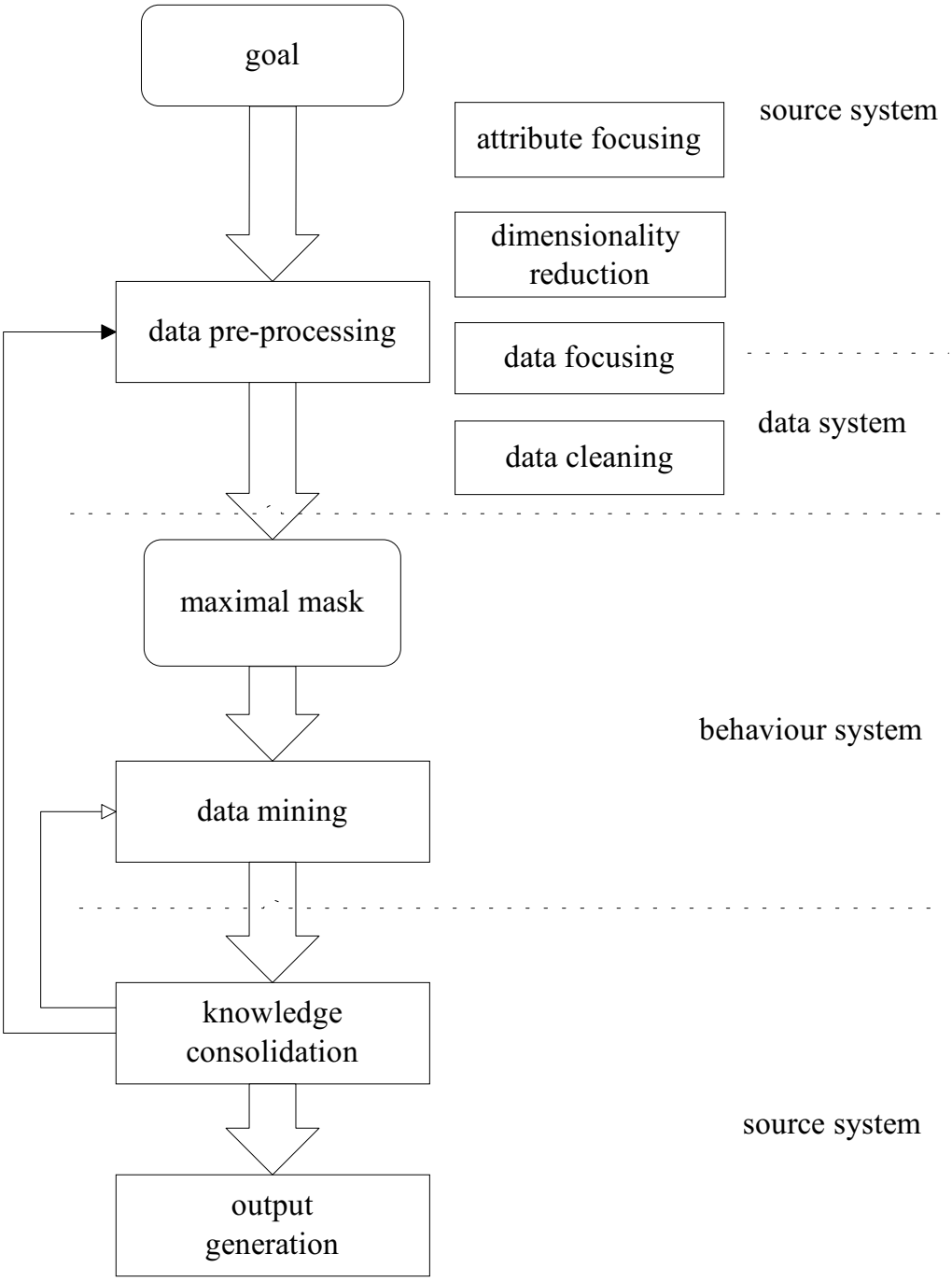


Figure 7.5 : GSPS's role in KDD

The data mining step in KDD belongs to the behavioural system. Before entering this level, the data is pre-processed, so one is left with the KDD task of finding an interesting time-invariant pattern in the pre-processed data. This time-invariant pattern is just that what is called a mask in GSPS.

A maximal allowable mask 'flattens' the data into a static matrix (such as the one in Table 7.4). On these 'flattened' data, many classification methods stemming from the KDD application domain are applicable. Therefore, the GSPS method can now borrow concepts and methods from three major scientific domains as illustrated in Figure 7.6.

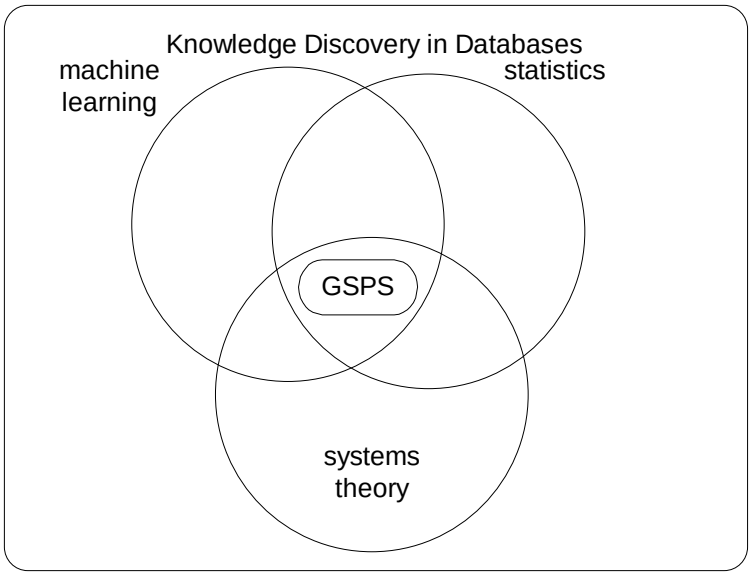


Figure 7.6 : SAPS and KDD

It is out of the scope of this thesis to explain all data mining methods in detail. Consequently, only an overview of the ones that are readily applicable will be given. In addition, the specific domain these methods are situated in, is mentioned.

- Rule-learning: rule learning techniques, which originate from the field of machine learning, can be used. Rules are easy to understand for domain experts, and they can be easily coded into a rule base of an expert system. A nice and very desirable side effect is that a rule base of dynamical behaviour can be automatically built. In chapter 8, a way for automatic generation of rules (via tree classifiers) will be illustrated. It is not the only way: other packages exist that find rules straight from the data and not via a tree classifier, see [Segal and Etzioni 1994].
- Tree classifiers are good candidate models [Van Welden 1998]. They adhere to what already has been done with SAPS (e.g. the hill climbing approach). They also show the links with the domain of machine learning and the domain of statistics in an explicit way. Tree classifiers and regression trees constitute a major part of chapter 8.
- Neural networks are well adapted to find patterns in data very fast. However, tractability (and the corresponding legal) issues may exclude their use in certain applications. A description of a tool that uses neural networks (among others) is found in [Khabaza and Shearer 1995].
- Regression analysis: the state components can be considered as regressors and the output as the dependent variable. Of course, distributional assumptions are quite unlikely to be fulfilled in general. Hence, the use of non-parametric regression techniques may be required.
- Genetic algorithms are other candidate techniques, but due to the non-positional importance of a state, techniques like crossover are unlikely to result in a good classification, [Goldberg 1989]. Mutation may play a more important role; hence, the use of evolutionary algorithms may prove to be more beneficial.
- Non-parametric discriminant analysis is another technique that may be applied. Usually, a multivariate normal distribution cannot be assumed to hold. Of course, when normality

assumptions prove to be applicable, a corresponding parametric technique should be applied to achieve better results.

- The nearest neighbours method is in essence a non-parametric technique. It is based on proximity measures, and when recoding is not used, it may be applied (note the relationship with the discussion of the influence of the type of variables on certain classification methods). In chapter 8, this method will be used and compared with regression trees.
- When recoding is used, one may end up with ordinal or nominal variables. Case-based reasoning may then be more appropriate because it can deal with lower type variables.

The summary above is not meant to be complete. Its only purpose is to show the enormous area of further research about the applicability of existing techniques in other domains of science that now become available (e.g., other paradigms like Bayesian networks may be well applicable).

The consequences of using a data mining approach to GSPS are thus numerous. They will be highlighted in the conclusion part of this thesis, because they open the field for a whole new range of research that can be done.

7.4 Advantages and disadvantages of data mining approach

There are many advantages by applying the data mining approach.

+ Recoding can be handled automatically by some data mining methods: tree classifiers will illustrate this in detail in chapter 8. Any kind of data can now be tackled, because some methods from the field of KDD that are well equipped for handling certain variable types, can now be applied. A plethora of different methods for different circumstances, e.g., type of variables, distributional assumptions, etc., is now available.

+ Databases can now be large, both in the number of variables and in the number of records. Putting derivatives in the raw data matrix allows detecting more relationships. One can obtain this by differencing. The latter method is also used in time series to remove the effect of a trend. Hence, taking the first order (or higher) differences may reveal interesting patterns in the data, but at the expense of more computational power. Fortunately, data mining methods are more suited to handle a large number of variables (and records). An example of differencing and trend removal appears in chapter 8.

+ Prediction models can be built that are more comprehensible or more accurate. Rule-bases can be automatically generated. Again, this will be illustrated in chapter 8 by the use of examples.

+ The concept of a training and test set is now automatically applicable, because it is a fundamental concept of the data-mining field. Extra validation tools, such a cross-validation, can now be used for smaller data sets. Chapter 8 illustrates this via examples, where cross-validation for accessing the (internal) validity of the generated prediction model is used

+ Missing values can be automatically handled by some methods. Examples will be given when tree classifiers are discussed in more detail in chapter 8.

+ Cleaning of data is also a fundamental concept in data mining. There was not enough emphasis placed on this process in the field of system identification approached such as GSPS. Now it becomes an integral part of the whole approach.

+ A plethora of techniques now becomes available, and thus many different methods can be compared with each other and validated. Hence, the methods themselves are evaluated too. This is also comprised in the data mining approach.

There is one major 'drawback'.

– The only major 'disadvantage' lies in the complexity of the approach. The newly added complexity of the data-mining approach with its plethora of techniques must be kept under control. The multi-paradigm approach requires knowledge of both domains, which is not easy to obtain[6] when one knows that GST and KDD themselves constitute quite some knowledge.

7.5 Resulting paradigm shift for SAPS

GSPS is related to KDD. More specifically, the bottom-up approach in GSPS corresponds with the knowledge discovery approach in KDD. The latter corresponds with inductive learning (in ML). SAPS works with directed systems and one-output masks. Hence, while GSPS was still related to inductive learning, SAPS restricts itself to supervised learning.

Consequently, SAPS can be enhanced with classification and regression methods after flattening the raw data via a maximal allowable complex mask. SAPS inherits all advantages from the data-mining approach to GSPS. As an extra advantage, (pre-)recoding (which may result in a worse system identification) is not necessary anymore (it can be handled more optimally), and metrics can be kept. An example of this is found in chapter 8, in which a new and simplified nearest neighbour method is evaluated.

Application of classification and regression techniques in the context of SAPS, constitutes quite an area of research for which the newly established links with KDD provide the necessary feed for further investigations, (see Figure 7.6). Many of these techniques and their performance are still under investigation. Figure 7.7 contrasts the data mining approach to the approaches in SAPS-II and SAPS-ST.

The exhaustive mask search in SAPS-II has as the obvious advantage that it will always find the 'best' mask (according to the chosen criteria). However, the search space can become very large because of the high cardinality of a maximal allowable mask. The depth of this mask cannot be taken too deep, because the exhaustive search limits the number of state variables that can be used in the optimal mask searching. The computational complexity limits the applicability of the exhaustive search technique to data with not many variables (typically less then 10) and to restrictive mask depths. So, one could ask the question if, in these circumstances, the optimal mask can be found (is it not out of range of the possibilities of the search?). Alternatively, one could ask if exhaustive search is still feasible for real complex systems?

[6] Personal experience of the author.

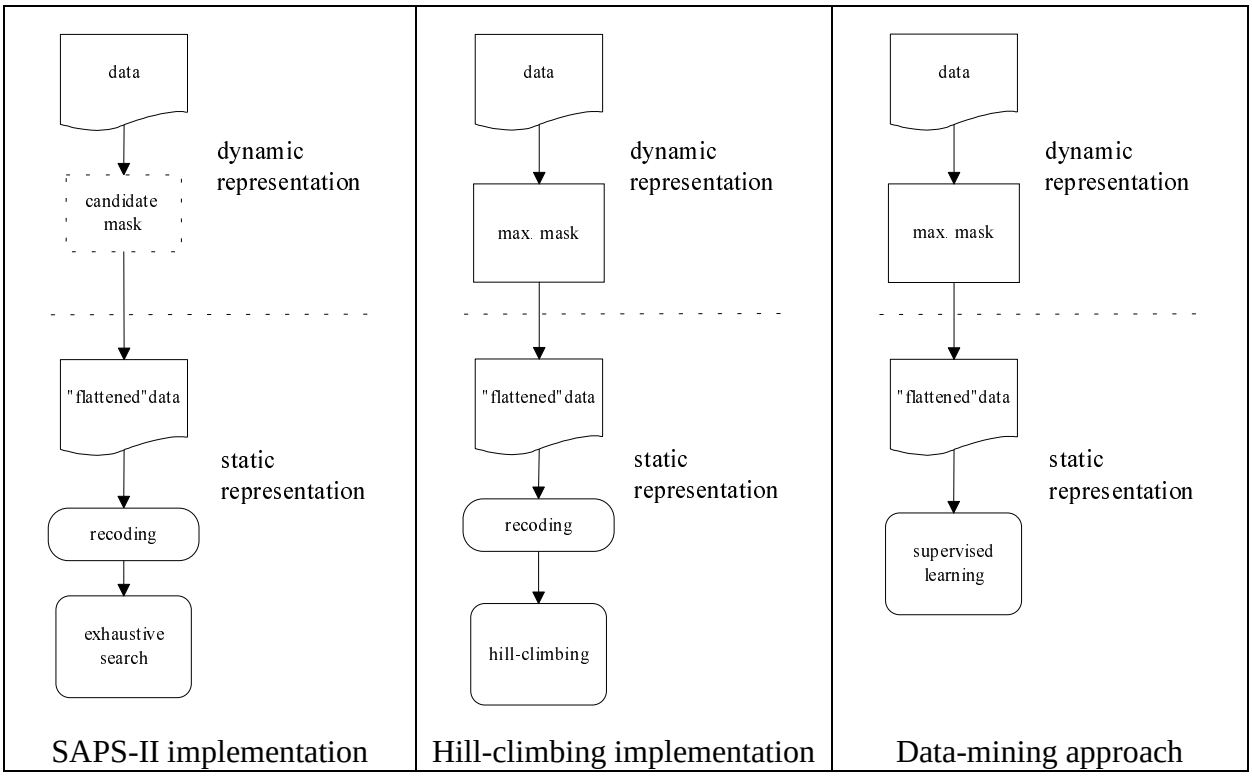


Figure 7.7 : Transforming I/O behaviour into static records

The hill-climbing approach allows tackling more complex systems than SAPS-II can. It puts the different patterns in relation with each other via a tree structure. It emphasises the concept of a *maximal allowable mask* (as constraint and as starting point) and it introduces a *structure* in the hypothesis space. The depth of the mask could be set much higher now, because of the new searching algorithm. Hence, masks of higher cardinality (deeper of wider) can be tackled. However, the global optimum may not be found, see chapter 4. The latter is noticed in many classification techniques also.

This concept of maximal allowable mask comes back in the data mining approach. The structure of the hypothesis is now determined by the model specification. The latter can be chosen such that very large data sets can be dealt with. This is much related to the scalability of the selected model. Scalability is a major issue in KDD. It will be shown that SAPS-II is not scalable, SAPS-ST can tackle larger data sets and it is more scalable. The new data mining approach is suitable for very large data sets. Data mining techniques exist for all kind of variables, so the data mining approach is quite powerful. This issue comes back in the final conclusion of this thesis.

7.6 Conclusions

The emphasis in GST is more on the dynamical structure of models, while KDD is more focused (until now) on the static structure of models (both terms, i.e., dynamical and static, are defined in [Ören 1984]).

It is demonstrated that modelling in its most general meaning consists of the same steps in both the domains of general systems theory and knowledge discovery in databases. Some paradigm shifts are revealed, but nothing prevents a better merging of both domains. KDD focuses a bit more on the comprehensibility and interestingness of a model, hence making it more general in that aspect. GST focuses more on time-dependent data and on model decomposition. For complex problems, the decomposition aspect can be used in KDD too. It may be

better to split a very complex system in sub-systems, and to tackle each subsystem in turn and then to cast the information found in a framework that keeps count of the coupling between the sub-systems.

The bottom-up approach in GSPS can be seen as a learning paradigm. The mapping relies on the transformation of dynamical to static data with the aid of a maximal allowable mask. The different data-mining methods to be applied should be preferably non-parametric in nature (in the statistical sense). Hence, tree classifiers, neural networks, non-parametric regression, genetic algorithms, non-parametric discriminant analysis, rule generators, nearest neighbours, etc., can now be applied to dynamical systems to find patterns in their behaviour. A merging of both domains seems feasible and the resulting meta-modelling methodology seems to be very powerful for both domains.

SAPS in related to supervised learning. Hence, the corresponding data mining approach carry over to SAPS. The tree classifier method forms a natural extension of the hill-climbing approach used in SAPS-ST. From the hill-climbing approach, the principle of a maximum allowable mask that represents the most complex allowable pattern is borrowed, and the exploitation of a tree structure to search for good sub masks (via a criterion) is utilised.

In this thesis, it is virtually impossible to demonstrate all data-mining methods, so we restrict our consideration to tree classifiers. It implies automatic rule generation. As a spin-off of the data mining approach, the nearest neighbours method used in SAPS-II will be shown in a new promising perspective.

References

Breiman L., Friedman J. H., Olshen R. A., Stone C. J. [1984], Classification and Regression Trees. Chapman & Hall, 1984.

Elzas M.S. [1984], "*System Paradigms as Reality Mappings*", Simulation and Model-Based Methodologies: An Integrative View, ed. Ören T.I., Zeigler B.P. and Elzas M.S., NATO ASI Series, Series F: Computer and System Sciences, vol. 10, Springer Verlag, p. 41- 68, 1984.

Friendly M. [1991], The SAS System for Statistical Graphics. Cary, NC: SAS Institute Inc, 1991.

Goldberg D. E. [1989], Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.

Han J. [1999], Personal communication, (from a lecture for the IBM chair in Antwerp 1999).

Karplus W.J. [1976], "*The Spectrum of Mathematical Modelling and Systems Simulation*", Simulation of Systems, ed. Dekker L., North-Holland Publishing Company, p. 5 – 13, 1976.

Khabaza, T., Shearer, C. [1995], Data Mining by Data Owners: Presenting Advanced Technology to Non-Technologists through the Clementine System, Intelligent Data Analysis '95, Baden-baden, 1995.

Klir G.J. [1969], An Approach to General System Theory. Van Nostrand Reinhold, 1969.

Klir G.J. [1985], Architecture of Systems Problem Solving. Plenum Press, 1985.

Minsky M.L. [1965], "*Matter, Mind, and Models*", Proceedings of the IFIP Congress, 1, Spartan Books, p. 45-49, 1965

Ören T.I. [1984], "*Model-Based Activities: A Paradigm Shift*", Simulation and Model-Based Methodologies: An Integrative View, ed. Ören T.I., Zeigler B.P. and Elzas M.S., NATO ASI Series, Series F: Computer and System Sciences, vol. 10, Springer Verlag, p. 3-40, 1984.

SAS [1999], see web page from SAS Institute, www.sas.com

[1994] Segal R., Etzioni O., "*Learning decision lists using homogeneous rules*", Proceeding of the 12th Nat. Conference on A.I., p. 619-625, 1994. More info on www.isl.co.uk

Van Welden D., Verweij D., Vansteenkiste G.C. [1991], "*A Proposal for Incorporating Heuristic Knowledge in a Multifaceted System*", Proceedings of EUROCAST 91 - 2nd International Workshop on Computer Aided Systems Theory, Krems (Wachau), Austria, April 15-19, p. 295-306, 1991.

Van Welden D., Vansteenkiste G.C. [1992], "*A Mixed Deductive-Inductive Approach to Model Recognition*", Proceedings of the 1992 European Simulation Multiconference, York, UK, June 1-3, p. 112-118, 1992.

Van Welden D., Kerckhoffs E.J.H., Vansteenkiste G.C. [1998], "*Extending a Fuzzy Inductive Reasoner with Classification Procedures*", Simulation Technology: Science and Art, Proceeding of the 10th European Simulation Symposium and Exhibition, ESS 98, ed. A. Bargiela and E. Kerckhoffs, October 26-28, Nottingham, UK, p. 111- 116, 1998.

Van Welden D. [1998], Tree Classifiers as Data Mining Tools, MSc. thesis, Catholic University of Leuven, Belgium, 1998.

Zeigler B.P. [1976], Theory of Modelling and Simulation. John Wiley & Sons, 1976.

Zeigler B.P. [1984], "*System Theoretic Foundations for Modelling and Simulation*", Simulation and Model-Based Methodologies: An Integrative View, ed. Ören T.I., Zeigler B.P. and Elzas M.S., NATO ASI Series, Series F: Computer and System Sciences, vol. 10, Springer Verlag, p. 91-118, 1984.

Chapter 8

The Use of Classification and Regression Trees for SAPS

8.1 Introduction

This chapter demonstrates the data mining approach, which was introduced in chapter 7, by showing how tree classifiers can be used in SAPS. Four examples, two synthetic and two real-world are worked out in more detail in the appendices. They illustrate different aspects of the tree classifier approach to SAPS. It will be established that the application of tree classifiers offers some significant benefits compared to the approach in SAPS. For example, the subjective way of recoding in SAPS is resolved via dynamic discretisation, and missing values can now easily be handled.

Eager learning via tree classifiers is contrasted with lazy learning via the nearest neighbour method in most appendix examples. A spin-off result of the data mining approach concerns automatic rule-generation. Tree classifiers can also do feature selection, which is a necessary step for the nearest neighbour algorithms that follow. Moreover, the selected features can be fed back to SAPS to construct a new primary mask.

Finally, an alternative five nearest neighbours method is introduced and compared with the existing method in SAPS-II. In addition, a further comparison is done with the introduced tree classifier approach.

8.2 Recoding and quantisation issues

The qualitative aspect of SAPS arises from (re)coding of all variables (inputs and outputs) into lower type variables. A taxonomy of variable types is depicted in Figure 8.1. Variables at the top are more mathematically 'rich' than variables at the bottom, in the sense that higher type variables[1] contain more 'information' in their levels than lower type variables. For example, nominal variables are categorical variables that have no natural ordering, while ordinal/rank variables do possess an ordering, but distances between the levels are unknown or not applicable. Only for interval variables and higher types, one has a distance measure. All operations for variable types with a lower 'information content' work for variables with a higher 'information content', while the inverse is not true. For a more detailed description, see [Van Welden 1998].

Evidently, recoding gives information loss, and thus it would be nice if one has at least the possibility to look for patterns in data without recoding first. The loss of accuracy is not, how-

[1] It is better to use "higher type" than "higher level", because the latter here has another meaning in this thesis (e.g. level of an attribute)

ever, in forecasting, because the use of fuzzy measures allows quantitative predictions by using membership and side values. The information loss is thus in the recoding itself and the way in which it is done. The number of levels is determined by a rule of thumb (chapter 2), which renders recoding quite subjective: one could ask if the recoding is optimal. A better solution would be to include the recoding in the data mining method itself.

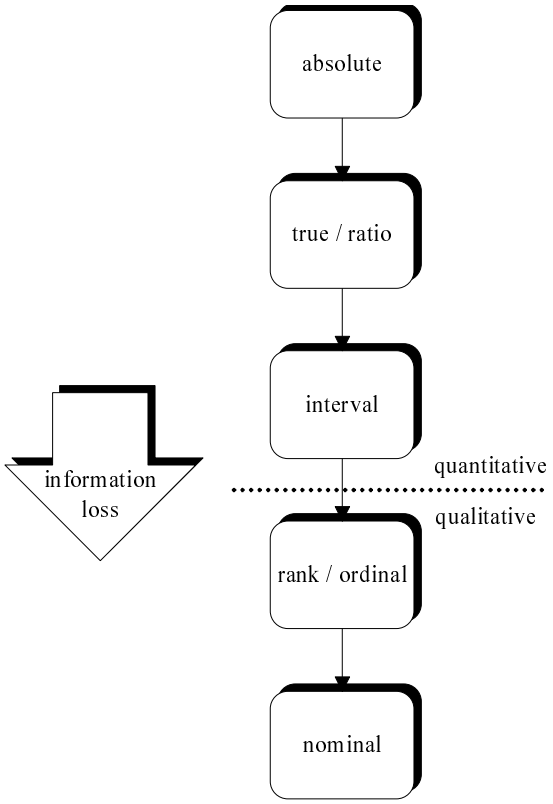


Figure 8.1 : Variable types

Chapter 2 explains two recoding techniques (equidistant or uniform recoding) in SAPS, which can be used to obtain good categorisations (statistical term) or quantisations (system-theoretic term). However, it is important to remark that recoding can break a metric associated with the raw data types. For example, under uniform recoding, the intervals may have not the same width and thus, although ordinality is kept (ordinal variables), there are no interval variables anymore. This may have negative consequences for the data-mining methods to be applied later. If recoding has to happen anyway it is better that recoding is postponed as much as possible and that it keeps as much relevant information as it can without excessive expressiveness. Section 8.2.2 shows how this can be done with tree classifiers.

8.2.1 Known quantisation methods in KDD that are applicable to SAPS

The most straightforward discretisation method in KDD is equal-width (distance) partitioning, which divides the range into N intervals of equal size (uniform grid). If x_1 and x_n are the lowest and highest values of an attribute, the width of intervals will be: $w = (x_n\text{-}x_1)/N$. It has some disadvantages, such as the fact that outliers may dominate, and that skewed data is not handled well. This quantisation is already being used in SAPS, and one has the additional drawback that it results in a considerable knowledge loss.

Newer in KDD is equal-depth (frequency) partitioning, which divides the range into N intervals, each containing approximately the same number of samples. This gives better data scal-

ing, (see chapter 2), but managing categorical attributes can be tricky. The latter problem is of less importance in SAPS, where attributes are usually continuous.

Finally, two other discretisation techniques may come into play. The first is Chi-Merge, [Kerber 1992], which is a quantisation technique that relies on a chi-square test to discretise numeric attributes repeatedly. Chi-Merge's basic principle is that relative class frequencies should be fairly consistent within an interval (otherwise they should split), and two adjacent intervals should not have similar relative class frequencies (otherwise they should merge). It is implemented via a statistical measure, the χ^2 test, to test the hypothesis that two discrete attributes are statistically independent. For two intervals, if the test concludes that the class is independent of the intervals, the intervals should be merged. If the test concludes that they are not independent, i.e., the difference in relative class frequency is statistically significant, the two intervals should remain separate. Hence, the algorithm consists of two steps:

1. Compute the χ^2 value for each pair of adjacent intervals
2. Merge the pair of adjacent intervals with the lowest χ^2 value

Repeat (1) and (2) until the χ^2 values of all adjacent pairs exceeds a threshold, which is determined by the significance level.

The second discretisation method known in KDD is based on entropy. A given set of samples S is partitioned into two intervals S_1 and S_2 using a boundary T. The entropy H after partitioning is (# denotes the cardinality of the sample)

$$H(S \mid T) = \frac{\#S_1}{\#S} H(S_1) + \frac{\#S_2}{\#S} H(S_2)$$

The boundary that minimises the entropy function over all possible boundaries is selected as a binary discretisation. The process is recursively applied to partitions obtained until some stopping criterion is met, e.g., $H(S) - H(S \mid T) > \delta$. Experiments show that it may reduce data size and improve classification accuracy. However, a better quantisation method is presented in the next section.

8.2.2 Dynamic quantisation for SAPS

Some classification algorithms have a built-in mechanism to discretise continuous attributes. The two 'classical' tree classifiers that were discussed in chapter 6 do possess that possibility. During the growth of the tree, an attribute is split binary[2]. In that process, many splits are tried and the best split chosen to decompose the node into child nodes (see equation 6.2). A similar thing happens for oblique trees, where a combination of variables is taken. Consequently, the partitioning of the variables into distinct level ranges is done in a dynamical manner. The levels do not necessarily have the same width or the same number of points. One could say that it is a kind of entropy based splitting (for C4.5) or a kind of misclassification probability splitting (for CART). Therefore, tree classifiers provide a powerful mean to quantise continuous variables. For categorical variables, a grouping is done to obtain the binary split. Evidently, this way of obtaining splits is clearly superior to the quantisation schemes available in SAPS. The attribute levels are not equal-width or uniform distributed, nor does each attribute have the same number of levels. Hence, the benefits are threefold:

- Recoding is done automatically and driven by the data in the learning sample,

[2] This is not entirely true in C4.5 for categorical variables, but it can be ignored because CART will be used in the sequel.

- Each attribute can have a different number of levels,
- The recoding scheme is more flexible because the width of each level is tailored on the pattern found (each level in an attribute can have a different width).

The power of quantisation by tree-classifiers can also be noticed from the numerous applications of tree classifiers for feature selection (e.g. see [Quinlan 1993 ; Breiman 1984]). This feature selection property is exactly what is desired for SAPS, because a good mask is nothing more than a matrix representation of which variables are deemed important for the predictive model (state-observation matrix) or 5NN method.

Hence, tree classifiers do an excellent job in selecting the important state variables for prediction. Consequently, they can be used in two modes:

1) use them to grow a tree and utilise that tree as a predictive model. This is an eager method.
2) use them to do feature selection only, and employ another, but lazy, method for prediction

In the first case, a tree classifier or regression tree is used to select the important variables and to predict as well. C4.5 and CART have this ability built in. An example of an induced tree classifier, which corresponds with the qualitative function

$$y(i) = \tilde{f}(u_1(i), u_1(i-1), u_2(i), u_3(i-3), y(i-1))$$

or with the mask (suppose there are five inputs) given in Table 8.1, is depicted in Figure 8.2.

u_1	u_2	u_3	u_4	u_5	y
0	0	-1	0	0	0
0	0	0	0	0	0
-1	0	0	0	0	-1
-1	-1	0	0	0	1

Table 8.1 : Mask for $y(i) = \tilde{f}(u_1(i), u_1(i-1), u_2(i), u_3(i-3), y(i-1))$

In the second case, a tree classifier is used only to pick out the most important variables. These are represented by the non-zero entries in a mask. A mask gives a good visual picture of the feature selection, e.g., see Table 8.1. This principle is already used in SAPS-II and SAPS-ST as a prerequisite for the 5NN prediction. CART adds to this that it can set up a ranking of variables via surrogate variables. The latter have a similar function as the utilisation of secondary masks for prediction purposes (section 2.3.4). So, one gets a battery of masks that do the necessary feature selection for a lazy algorithm such as 5NN (5 Nearest Neighbours), see section 8.4.1.

Performance[3] comparisons, between the eager and lazy case, are found in the appendices and in section 8.5. In the lazy case, predictions that are more precise may result.

[3] With 'performance' is meant 'accuracy in prediction'

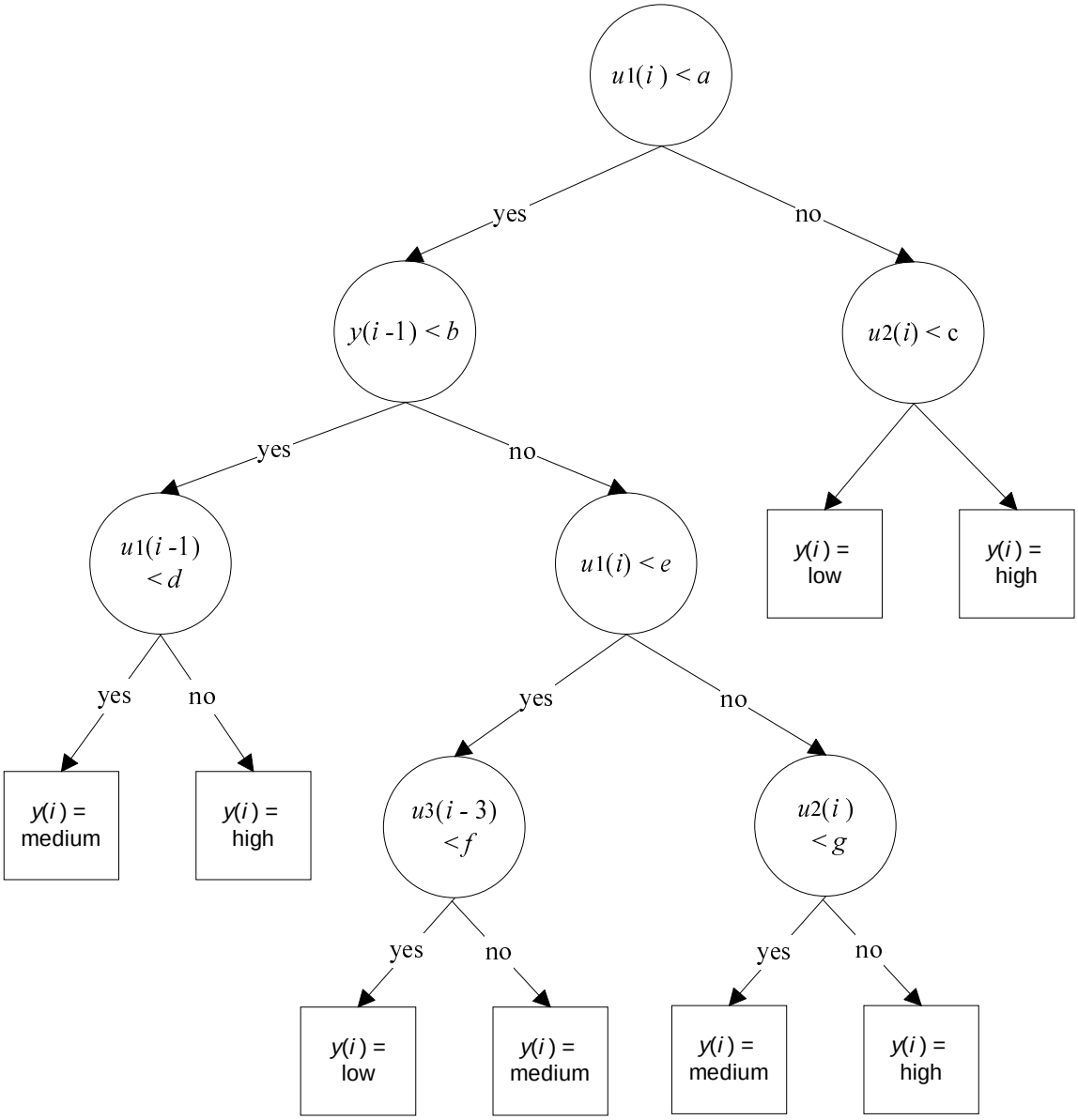


Figure 8.2 : SAPS example of tree classifier

In the conclusion of chapter 6, a motivation for using CART over C4.5 is given. In all examples in the appendices, the response is continuous. This is often the case for dynamic systems. Therefore, the primary motive for selecting CART over C4.5, is the requirement for regression trees. Regression trees do not require a priori recoding of the output: the latter is quantised during the tree induction.

In addition, surrogate splits, which are only available in CART, form the backbone in handling missing data, and they give a ranking of what are supposed to be the most important variables. This is better than simple feature selection. It also gives an idea of the importance of a feature, which is put on a scale from 0 to 100. Surrogate variables can be put in a separate mask. Hence, the concept of surrogate variables allows us to use a battery of predictive models for forecasting.

8.3 Machine learning approaches to SAPS

The machine learning approach to SAPS is the most obvious to begin with, because it gives the right focus towards understandable models, it attaches great importance to rules, and it emphasises algorithmic performance. The ID3 algorithm, which is introduced in chapter 6, is a relatively simple algorithm to start the comparison with SAPS-ST. The latter is not done with SAPS-II because SAPS-II does not use any tree structure.

8.3.1 Comparison of the ID3 algorithm and the SAPS-ST algorithm

The ID3 algorithm is a tree induction program, which is based on forward pruning and which uses a splitting rule based on the Shannon entropy, see section 6.8.1. Comparing this algorithm with the way SAPS-ST does sub-optimal mask searching, large similarities are perceived.

In SAPS-ST, a tree is grown from the root until some stopping criterion is triggered. The stopping criterion is based on a decrease in mask quality. When the best node quality on a level in the tree is lower than the current maximum quality, the expansion stops and the best node is retained as a sub-optimal mask. The degree of determinism (or 'purity') of a node in SAPS-ST is based on the Shannon entropy. Hence, if only entropy would be used as quality measure, one has a similar kind of tree induction as in ID3: both use a threshold decrement rule, which looks at the *decrease* in 'quality', and both algorithms are greedy. However, the 'quality' used in a split is based on a mixture of normalised entropy and complexity in SAPS-ST, and on information gain in ID3. Thus, there is some difference in the evaluation of node, but both methods look very much alike. The similarities between SAPS-ST and ID3 are made explicit in Table 8.2.

	SAPS-ST	**ID3**	**C4.5**
Structure	tree	tree	tree
Purity	distribution based (entropy)	distribution based (entropy)	distribution based (entropy)
Pruning	forward	forward	backward
Splitting/ grow	greedy	greedy	greedy
Node evaluation	quality based on entropy and complexity	information gain based on entropy	gain ratio based on entropy and complexity
Stopping	threshold decrement rule on quality	threshold decrement rule on information gain	-
Predictors	rule-of-thumb quantisation of continuous variables and categorical	categorical variables	automatic quantisation of continuous variables and categorical

Table 8.2 : Comparison of SAPS-ST, ID3 and C4.5

Chapter 6 showed that the splitting rule is quite important in *forward* pruning, so it must be based on a good heuristic. Therefore, the quality definition in SAPS-ST and the information gain definition in ID3 are crucial. The disadvantage of ID3 in favouring splits with many outcomes does not apply to SAPS, because the complexity component in the evaluation function

will penalise this kind of splitting. In that aspect, SAPS more resembles C4.5, which uses a better splitting rule to cope with this shortcoming.

The above considerations, together with Table 8.2, would lead to the conclusion that ID3 and SAPS-ST are very much alike. This is true from an algorithm viewpoint, but not from a conceptual viewpoint: the content of the nodes is different. SAPS-ST has masks in its nodes, while ID3 has data distributions in its nodes.

In ID3, each node corresponds to a non-goal attribute (predictor) and each arc to a possible value of that attribute, i.e., one is working in the measurement space. A leaf specifies the expected value of the goal attribute (response) for the records described by the path from the root to that leaf.

In SAPS-ST, each node corresponds to a mask and each arc to a mask-submask relationship, i.e., one is now in the model space. It is thus *not* so that the path from the root gives a classification for an outcome. Therefore, the use of the tree structure and the forward threshold-based pruning is not done on the same data structure. Consequently, paths from the root should not be interpreted as rules for classification.

The extensions and improvements that Quinlan made by going from ID3 to C4.5 do not make sense for the sub-optimal mask algorithm, except when the model search itself is considered important (meta-modelling). If the model search itself is to be modelled, then one can expand the tree slightly further to have an 'overfit' tree of *models*. 'Overfit' in this context means exactly the opposite as what is normally meant in tree classifiers, i.e., down the tree, models are less complex and thus underfit the data. With regard to that aspect, it would be better to compare SAPS-II with ID3, because both go from underfit to overfit models. Pruning back in SAPS-ST yields more complex models, and this is what one usually wants to avoid (Occam's razor). However, around a certain 'optimal' expansion of the tree (where the hill-climbing algorithm stops), it may be worthwhile to evaluate a 'band' of models on a test set to pick the best one. Such an approach is already found in chapter 4 (see Table 4.2) for an optimal mask search.

The situation becomes different when applying ID3 on the transformed data as described in chapter 7. In that case, one can apply any supervised data mining algorithm for SAPS. ID3 is just one of them. Nonetheless, there are better algorithms, such as C4.5 (or See5) to induce a tree classifier on the transformed data.

8.3.2 Using C4.5 for SAPS

The use of C4.5 for SAPS is nothing more than a concrete type of classifier, which is used on the static data generated by the 'flattening' transformation described in chapter 7. The advantages with regard to ID3 are stated in section 6.8.2. The reasons why CART is chosen instead of C4.5, are listed in 8.2.2.

The successor of C4.5, called See5, has some extra advantages: it allows boosting, it incorporates several new facilities such as variable misclassification costs, and it uses 'soften' thresholds [Rulequest 1999]. These enhancements will not be investigated further in this thesis.

8.4 Forecasting in SAPS via 'Nearest Neighbours'

The nearest neighbour and the locally weighted regression method are prototypes of an example-based method (see chapter 5). They can model very complex target functions by local approximations. One drawback of example-based methods is that computing takes place during classification of new instances itself, raising the computational cost of predicting. Another drawback is that they tend to use all attributes when looking for similar instances in the training set. However, the role of a mask in SAPS is to perform the necessary feature selection required for subsequent example-based forecasting. Thus, the problem of using irrelevant features is avoided.

8.4.1 Introduction of two new nearest neighbour methods

In SAPS-II, the five nearest neighbours method is used for forecasting. Section 2.3.4 explains that the closest point is used to forecast the class and side value of the new output, (see Figure 8.3), while the membership value is determined by the five nearest neighbours.

Referring to [Mitchell 1997], one should consider this as a mixture of 1 Nearest Neighbour (for class and side values) and distance-weighted 5 Nearest Neighbour for real-valued target functions. Consequently, in this thesis, this method is referred to as 5+1NN. One could as well apply the remark made by Mitchell to allow all training examples in the computation of the latter to have a global method instead of a local method. However, there are some complications involved, which are already present in the five first neighbours. The first is how exactly to determine the relative weights (see section 2.3.4). This discussion is not a major issue in this thesis.

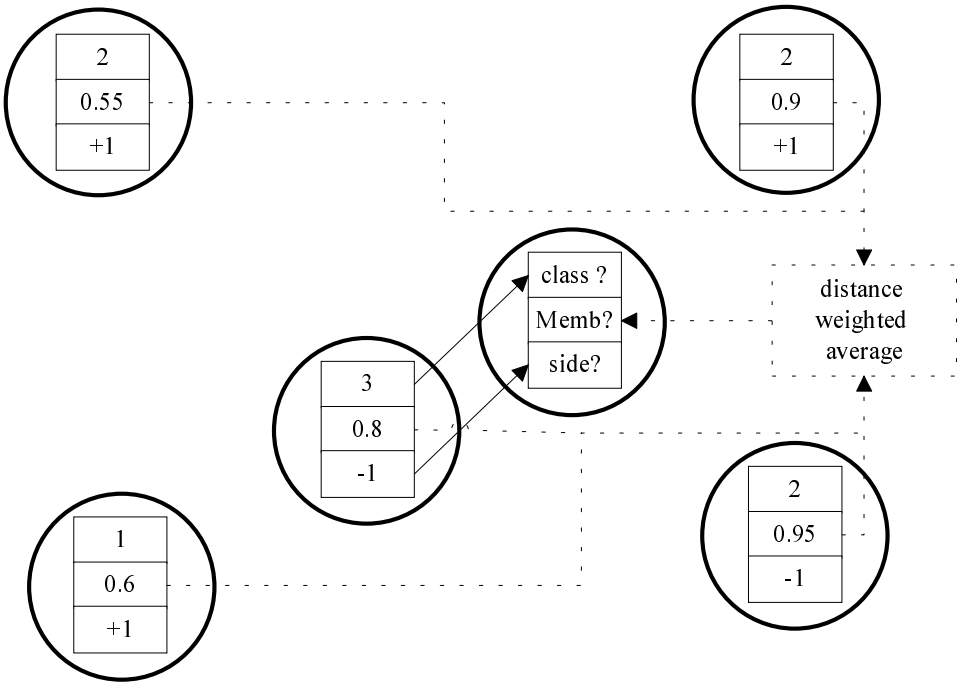


Figure 8.3 : 1+5NN algorithm in SAPS

A second, but more essential issue is that some nearest neighbours may lay in different recoded levels, giving another class and side value and thus raising the need to adjust membership values to allow a sensible computation of (2.20). This situation is depicted in Figure 8.3

where the closest neighbour has a set value of 3 and a side value of -1 (all values are for outputs in the static matrix, i.e. out_i). Consequently, the new output will be assigned set value 3 and side value -1. The membership value of the new output will be the distance weighted sum of all five neighbours, where care has to taken in combining membership values of neighbours with different set values. Besides, when looking at Figure 8.3, can one not take the set value 2 instead of 3 (three neighbours have set value 2)?

This methodological problem can be solved by a simplified five nearest neighbour method, which is proposed in this thesis. The new method relies on the raw transformed 'static' data for distance computing. Equation (2.18) can directly be applied to the raw data it can be applied to unit-rescaled data. The former method is denoted by 5NN and the latter by 5UNN. Both methods are implemented in SAPS-ST, together with the (5+1NN) method from SAPS-II. The (non-rescaled) 5NN method computes the distance between the points via

$$d(\vec{v}_i, \vec{v}) = \sqrt{\sum_{j=1}^{n} \left(v_j - v_{i,j} \right)^2} \qquad (8.1)$$

where $\vec{v}$ is the *m*-input vector of the current data record to be predicted (see equation 2.17), and $\vec{v}_i$ is the i^{th} *m*-input vector from the training set (see equation 2.16). However, one is left with possible scaling effects (on the relative weights) in which variables from a higher magnitude may contribute more than variables with a lower magnitude. Usually, this is not desired, so the 5UNN method, which uses a kind of normalisation, is often preferred. Unit-rescaling or normalisation[4] is obtained via

$$v_{scaled} = \frac{v - v_{\min}}{v_{\max} - v_{\min}}.$$

In determining the distances, values of the same variable are subtracted. This simplifies in equation (8.1) to

$$d(v_i, v) = \sqrt{\sum_{j=1}^{n} \left(\frac{v_j - v_{i,j}}{v_{\max,j} - v_{\min,j}} \right)^2} \qquad (8.2)$$

The determination of the relative weights is similar as for SAPS-II, but now the distances are computed by equation (8.2) for the 5UNN method or equation (8.1) for the 5NN method.

It is clear from a theoretical viewpoint that the procedure is conform to the distance-weighted *k*-Nearest Neighbour algorithm (here $k = 5$). Again, all training examples could be used to have a global method, but at the cost of more computing time, although much less severe than with the 5+1NN method. Furthermore, it is also conceptually sound because it uses immediate the raw (unit-rescaled) data. If the results are as good or better than what is obtained via recoded data (as done in SAPS-II), one has a quicker way to do forecasting, because much computational overhead can then be eliminated. This computational overhead concerns

- The recoding for the test set,
- The regeneration,
- The interpolation to get membership value right. This is theoretically difficult to justify.

[4] The term 'normalisation' can be used, but it is often associated (in the field of statistics) with making the mean 0 and the variance 1 for a distribution. Hence, it is preferred to use the term 'unit-rescaling'.

Appendix A demonstrates that the 5UNN method performs at least as good as the 5+1NN method. This is shown in Figure 8.4, which is in fact Figure A.4.

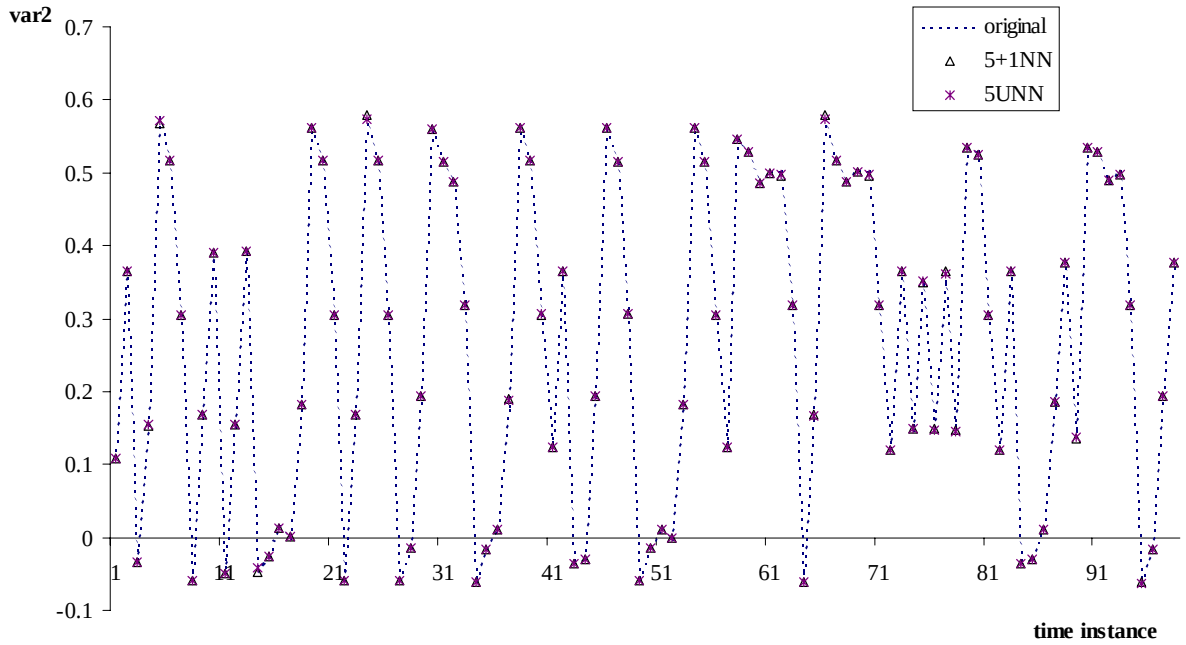


Figure 8.4 : Forecasting results under fixed recoding

The original (known) values for the test set are put on Figure 8.4, together with the forecasted values by the 5+1NN method and by the new 5UNN method. Both methods perform very well (see Figure A.6).

Table 8.3 shows clearly that the 5UNN method performs at least equally well as the 5+1NN method under the optimal mask. The table shows the Sum of Absolute Errors (SAE) and the Sum of Squared Errors (SSE) for the different forecasting methods.

	SSE	SAE
State-observation forecasting	1.53245	7.601
5+1NN (fixed recoding)	0.00061	0.117
5UNN (fixed recoding)	0.00040	0.105
5+1NN (uniform recoding)	0.00416	0.099
5UNN (uniform recoding)	0.00202	0.074
Regression tree	0.00096	0.213
Simplified regression tree	0.03666	1.486

Table 8.3 : Comparing different forecasting methods for the example in appendix A

Appendix B equally does a comparison between the 5+1NN method with the newly introduced 5(U)NN methods. The results are shown in Table 8.4 (Table B.8 in appendix B). It shows the much better accuracy of the predictions made with the 5(U)NN methods (both unit-rescaled and not). These numbers are confirmed by figures B.7, B.8 and B.10. Table 8.4 shows how the use of an optimal mask improves the forecasting.

	SSE	SAE
State-observation with sub-optimal mask	3485	455
5+1NN with sub-optimal mask	3723	369
5NN with sub-optimal mask	932	243
5UNN with sub-optimal mask	1011	249
5UNN with optimal mask	202	102
5UNN with second best mask	200	101
Regression tree	655	185

Table 8.4 : Comparing different forecasting methods for the example in appendix B

8.4.2 Comparing nearest neighbours with state-observation forecasting

The use of the nearest neighbour method, which relies on a metric, implies that only numeric attributes can be handled[5]. This restriction does not apply when a state-observation matrix is used. However, the prediction performance of the latter is quite bad. This is illustrated in appendix A by Figure A.13, which is reproduced here as Figure 8.5. It shows the results of an exhaustive mask search after which the 5+1NN method and the state-observation method are compared on a test set (together with the regression tree approach).

[5] However, Cost and Salzberg [1993] claim that the NN method can also be applied to non-numerical attributes. This issue is not investigated further here.

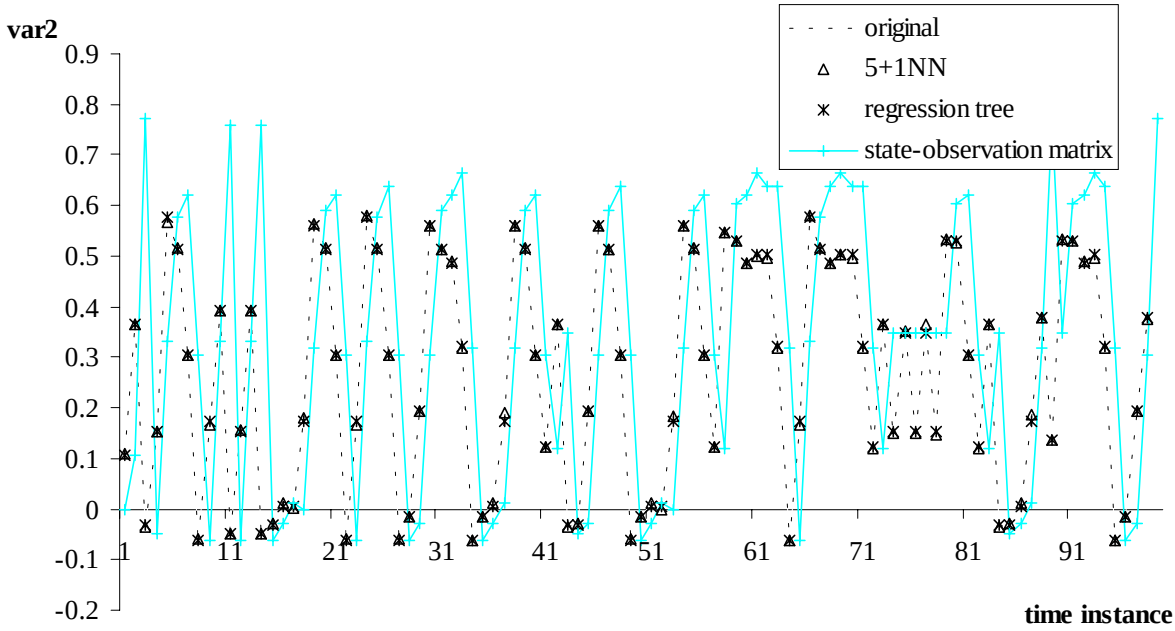


Figure 8.5 : Forecasting with 5+1NN, regression trees and state-observation method

From Figure 8.5, it is clear that the state-observation method performs less than the 5+1NN method. This is confirmed by Table 8.3. The example in appendix B (Figure B.6) shows a similar result. Inspecting Table 8.4 confirms the bad forecasting behaviour of the state-observation method, especially when compared to the newly introduced 5(U)NN methods. This is to be expected, because the latter are lazy methods that use the entire training set and that perform local fits. So, when a switch was made from state-observation matrix prediction [Cellier and Yandell 1986] to a nearest neighbour method [Cellier 1991], this was a good choice with regard to forecasting accuracy.

8.4.3 Conclusion about the new nearest neighbour methods

The 5(U)NN methods based on the static (raw) data are preferable from an algorithmic complexity viewpoint. It is also preferable from a conceptual viewpoint because it is simpler and theoretically sound. Finally, as the examples have shown, it is preferable because of forecasting performance with regard to accuracy. The new 5(U)NN methods perform at least as accurately as the original 5+1NN method, which is based on recoded values. Moreover, all this comes at a lower computational cost.

8.5 Advantages of tree classifiers for SAPS

+ Recoding for all kind of predictors can be handled automatically (dynamic discretisation). This is shown in all examples in the appendices.

+ In addition, no recoding is required for a continuous output when regression trees are used. This is the case for all examples in the appendices.

+ Rule sets can be generated automatically. This is shown explicitly in appendix A.

+ Tree classifiers can be used as prediction model (eager method) or they can serve as the necessary feature selectors for a lazy evaluation such as 5NN. A comparison of the (eager) regression tree method with the 5NN method is done in section 8.5. Section 8.6.4, which is based on appendix C, demonstrates how this can be done.

+ The sample size can vary and different techniques (test-train or cross validation) are tailored to this. The examples in appendices A, B and C use cross-validation, while the example in appendix D uses an internal test set.
+ Missing values can be handled. This is demonstrated in section 8.6.3, which is based on appendix C.
+ Additionally, CART ranks variables with regard to their importance. This is shown in all examples in the appendices.
+ Oblique trees with more expression power can be used (in CART). They were tried on the examples in the appendices, but they did not deliver better results. Other examples may benefit.
+ Many other tree classifiers can be applied that give a better performance or that have more expression power. This issue is touched in the conclusion of chapter 6.

8.6 Comparing regression tree performance with SAPS

In all examples in the appendices, the output is continuous. Hence, regression trees should preferably be used. Comparing the performance of regression trees with the nearest neighbour methods is always a bit unfair, because the latter are lazy methods that do a local fit on the whole training set. A regression tree is an eager method, which is more parsimonious, even when the tree is quite deep (it is still smaller than retaining the whole training set).

KDD uses an interestingness function for evaluating a model (or pattern). This can not always be formalised rigorously if aspects like comprehensibility come into play. Therefore, in the examples the option is taken to consider also much simpler regression trees with regard to their complexity (defined as the number of terminal leafs). Of course, the prediction accuracy will suffer from this, but the resulting rule base that can be generated automatically is more concise and thus better comprehensible. It is not the aim of the examples to do a rigorous investigation on complexity issues related with growing trees. This kind of research is the topic of more focused research. In this section, it is only the aim to demonstrate that regression trees do work: they can give good and yet parsimonious models.

The accuracy performance of a regression tree is measured on a separate test set for which the original values are known. Consequently, a basis of comparison is present. The accuracy is measured by looking at the Sum of Squared Errors (SSE) or by looking at the Sum of Absolute Errors (SAE), as was the case in section 8.4. These measures can always be applied regardless of model type or structure. Other measures of complexity are not used in comparisons, because they do not apply for different model paradigms. Regression tree performance will mostly be compared with the nearest neighbour methods, because it has been demonstrated in section 8.4.2 that the state-observation forecasting method delivers poor results in general.

For these comparisons one needs a (sub)optimal mask. Unfortunately, execution of an optimal or sub-optimal approach is not always possible. The reason for the former lies in the algorithmic complexity, while the reason for the latter concerns the practical implementation. Experience learnt that the sub-optimal approach, as it is encoded in the prototype, is too slow to cope with maximal allowable masks that have a high cardinality. One can call these kinds of masks entry-complex. Equation (4.2) can be used to compute the number of sub-masks that need to be evaluated. The number of negative entries (generating variables), denoted by *n*, is computed by $n = (m+1)*d + m$, where m is the number of inputs for the system and d the

memory depth of the mask. Equation (4.2) implies that the total number of masks to be evaluated is given by

$$T = \frac{n(n+1)}{2}$$

where n denotes the number of m-inputs, i.e. $n = \#M - 1$.

Table 8.5 shows the n values and the T values for various combinations of memory depth and number of system inputs (not m-inputs).

	n		T	
number of inputs $m \rightarrow$	$m = 3$	$m = 6$	$m = 3$	$m = 6$
memory depth d $\downarrow$				
$d = 12$	51	90	1326	4095
$d = 24$	99	174	4950	15225
$d = 36$	147	258	10878	33411
$d = 48$	195	342	19110	58653
$d = 60$	243	426	29646	90951
$d = 72$	291	510	42486	130305

Table 8.5 : The effect of entry-complexity in the sub-optimal approach

The evaluation of a mask involves determination of the Shannon Entropy (via a state-observation matrix) and the determination of a complexity value. By measuring the time to evaluate one mask, one can have an idea how long it would take to do a complete sub-optimal search. Timing was done on a PC (PII-333Mhz, 128 MB under NT4SP3). It shows that, when taking 6 inputs, one needs approximately 55 seconds for one mask evaluation if the memory depth is 12, 95 seconds if the memory depth is 24, and 132 seconds if the memory depth is 36. The corresponding estimates for total evaluation is then approximate 62 hours, 401 hours, and 1225 hours. If one takes only 3 inputs with a memory depth of 48, then the time to evaluate one mask is approximate 72 seconds, so a complete run would take approximate 382 hours. These numbers are only rough estimates because

- the evaluation time is not equal for all levels
- they depend heavily on the hardware configuration
- they depend if graphical output is desired at the same time or not (here not desired).
- swapping and garbage collection in the Smalltalk environment is not included in the estimates.

E.g., it took approximate 3 days to finish a run with 6 inputs and a memory depth of 12, which corresponds nicely with the estimate.

These considerations led to the conclusion that only regression trees (up to now) can tackle very (entry) complex masks. Consequently, this is the reason why in appendix C and D only regression trees are tried.

8.6.1 Applying regression trees to synthetic data sets

Mostly, an optimal regression tree is selected within one standard error (see section 6.9.2). Such a tree will be referred to as a 1SE tree. The 1SE tree for the example in appendix A is selected from Figure 8.6. It consists of 25 terminal nodes.

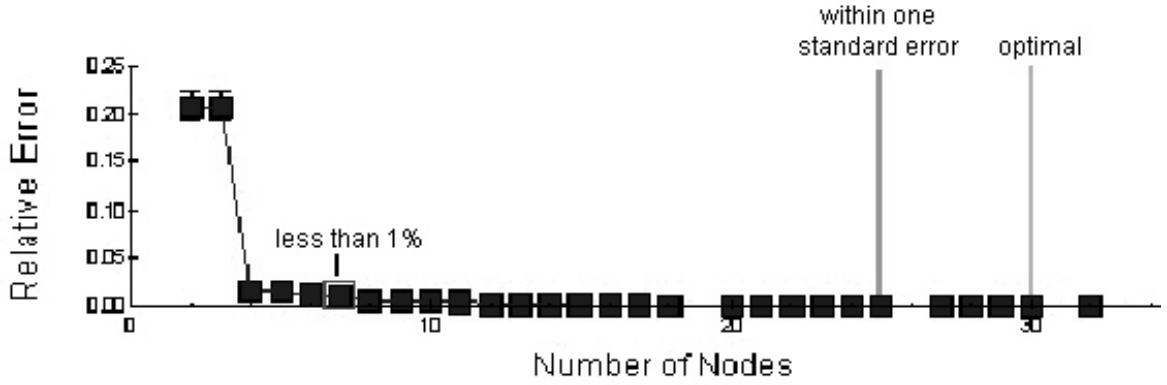


Figure 8.6 : Relative (cross-validated) error for the example in appendix A

The variables that are used for the best split are shown in the internal nodes of the tree in Figure 8.7. The notation that is generally used in the examples is that a variable *x*_1 stands for *x*(*i*-1) because indices cannot be used in CART. Therefore, a number appearing at the end of any variable always tells how many time steps this variable is situated back in time (_0 is not depicted).

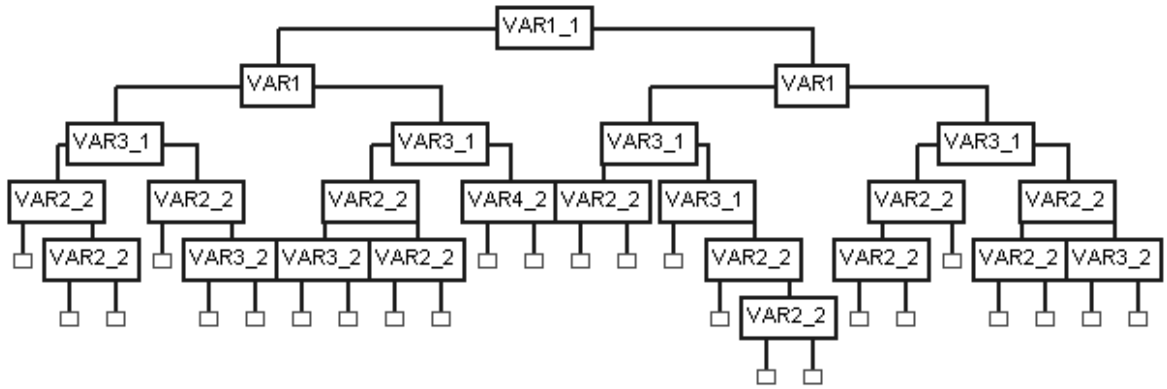


Figure 8.7 : The 1SE tree for the example in appendix A

Another plot, which may come handy, shows the box plots for the leaves sorted by target variable prediction. Such a plot is Figure 8.8. It shows the variability present in a leaf. The effect of this variability will re-appear in predictions.

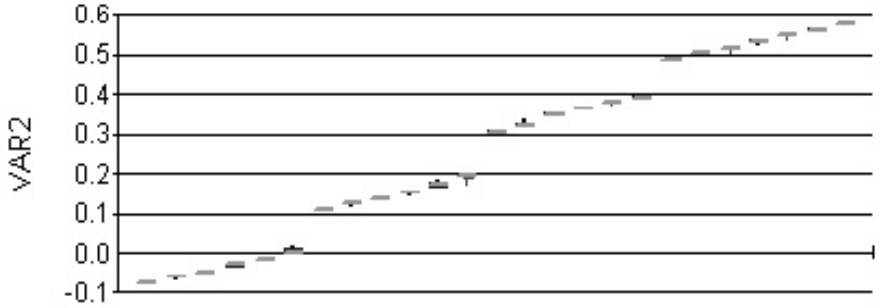


Figure 8.8 : Box plots for terminal nodes sorted by target variable prediction

Figure 8.5 displays the forecasting results from applying the regression tree in Figure 8.7 to a test set. It is hard to see any difference with the 5+1NN method. Table 8.3 confirms that the regression tree can compete with the 5+1NN method, despite its eager character. Regression trees (and classification trees) inherit all the advantages of eager methods.

- They are concise (compared to the data set a nearest neighbour method needs).
- They are much faster in prediction because their model is 'global' and fixed, once testing begins (eager method) and because they are more parsimonious in the sense just mentioned.

The good accuracy for the regression tree is to be expected when looking at Figure 8.8: notice the small variability in the terminal nodes. The conclusion is that the 1SE regression tree is to be preferred for the case in appendix A.

One could try a simpler regression tree. This is also done in Appendix A, where, if one takes the relative cross-validated error to be just under 1%, one obtains a '<1%' tree with only 7 leaves, see Figure 8.6. Figure 8.9 shows the (expected) larger variability in the leaves.

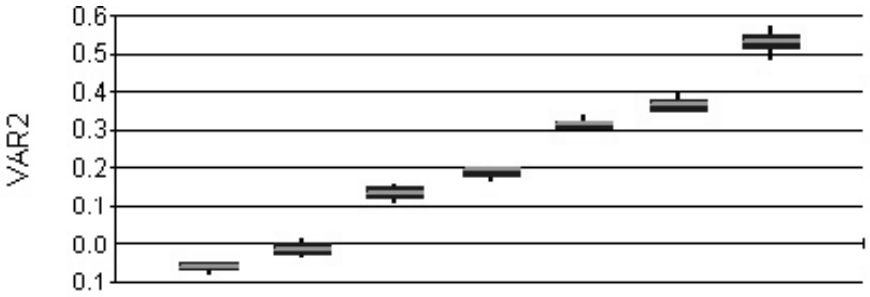


Figure 8.9 : Box plots for terminal nodes sorted by target variable prediction for the less-than-one percent tree

The forecasting is less accurate, which can be seen by comparing Figure 8.5 and Figure 8.10. Still, from a ML viewpoint, when one attaches a lot of importance to the lesser number of rules (only 7 rules), the resulting prediction is still quite well. The automatically generated rules are found in appendix A, Figure A.20.

Remark that the simplified regression tree still performs much better that the state-observation matrix prediction.

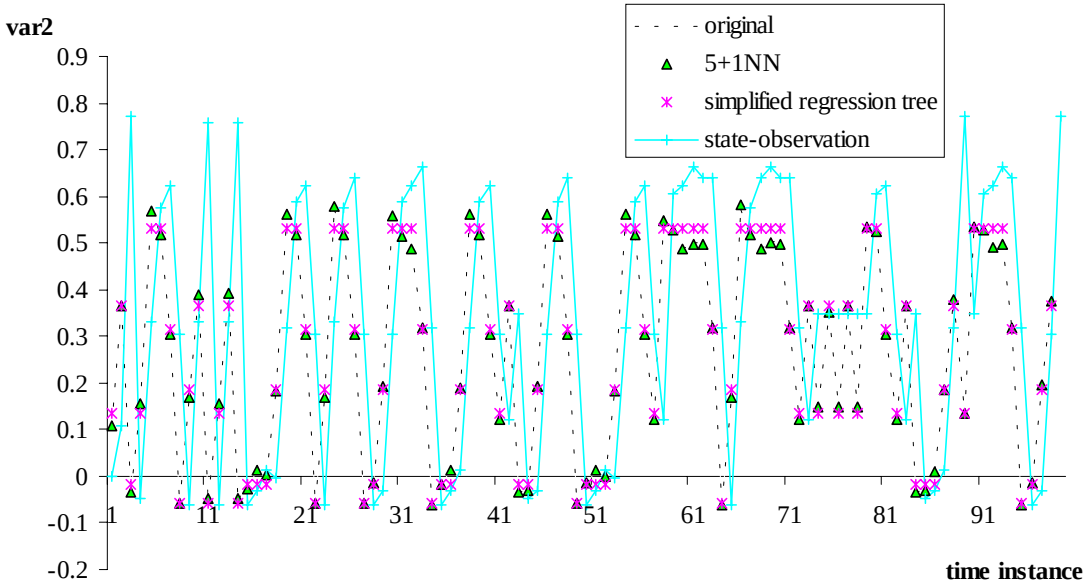


Figure 8.10 : Forecasting with 5NN, '<1%' trees and state-observation method

These findings are confirmed by Table 8.3. An interesting picture for comparisons can be found in Figure 8.11. It illustrates where deviations of predicted values (with regard to the known ones) occur.

Predicting with regression trees always imply 'activating' a terminal node. Hence, it is to be expected that forecasting is categorised by the terminal nodes. The corresponding predicted values are shown as horizontal lines on Figure 8.11. In this figure, the support (values that do occur in the test set) is indicated to explain why there are gaps in the plot. Additionally, Figure 8.11 reveals, in a graphical manner, why simpler regression trees give worse predictions in general.

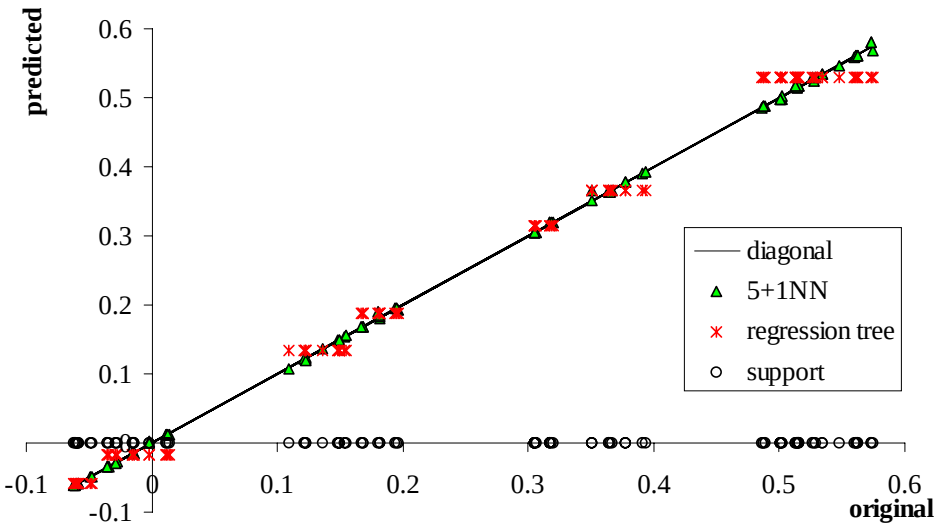


Figure 8.11 : Plot of predicted versus original data for the example in appendix A

Appendix B provides another comparison between regression tree forecasting and the 5UNN method. Figure 8.12 shows that regression tree predicts worse for small values of the target variable. For more details, the reader is referred to appendix B.

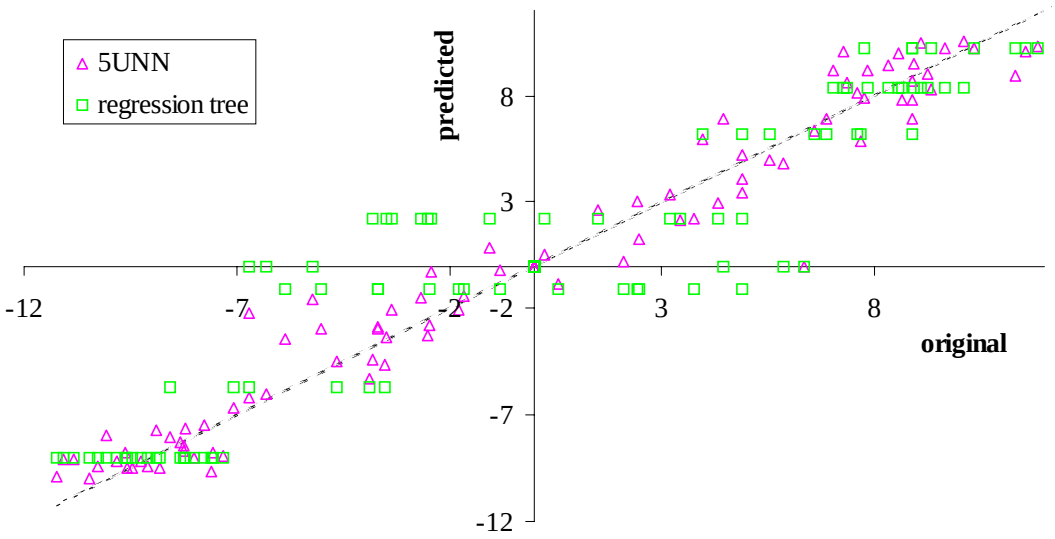


Figure 8.12 : Predicted versus original output values for the 5UNN method under the optimal mask and for the 1SE regression tree

Table 8.4 confirms the better performance of the 5UNN method, but also shows that it is not an order of magnitudes different. If one compares the regression tree performance with the nearest neighbour methods based on data obtained from a sub-optimal mask, then the former gives even better predictions. Hence, the worse prediction is only valid when comparing the (eager) regression tree method with the 5UNN (lazy) method used on data obtained from an *optimal* mask (which is sometimes impossible to obtain).

Therefore, this thesis claims that a regression tree provides a good predictive model, which can compete with the (lazy) nearest neighbour methods

8.6.2 Applying regression trees to real-world examples

Appendix C describes observations coming from a database of economic data, [Crombez 1999]. In this data set, two variables contain a trend. Hence, appendix C provides the opportunity to explore the pattern-recognition approach in the presence of a trend.

Appendix C demonstrates that SAPS' prediction *on a test set* fails completely (Figure C.3 in the appendix). This was expected due to the nature of the pattern recognition approach (something has to be observed to be able to be learnt). It is shown in appendix C that a regression tree induced on the same training set, was able to give a very good resubstitution forecasting, but its prediction also fails when predictions are done on a test set. This is observed in Figure C.5. This finding is not in contradiction with the principle that a good internal fit (interpolation) does not guarantee a good extrapolation. To be able to use regression trees (or SAPS), one has to do a fit on recurring patterns in the training set. For that to be possible, one must detrend the training set (and test set).

In methodological set-ups, it is always desirable to have a very general method for solving problems. Such a method is found by taking differences. Hence, the first thing tried was to differentiate the data by taking the first order differences. Unfortunately, it did not work out, neither for mask searching (Figure C.6), neither with regression trees (Figure C.8).

Because of the failure of the first detrending method, the data has been detrended 'manually' via a polynomial fit. It proved only necessary to detrend the variables MSC and IP. Their detrended curves are displayed in Figure 8.13 (which is Figure C.9 from appendix C).

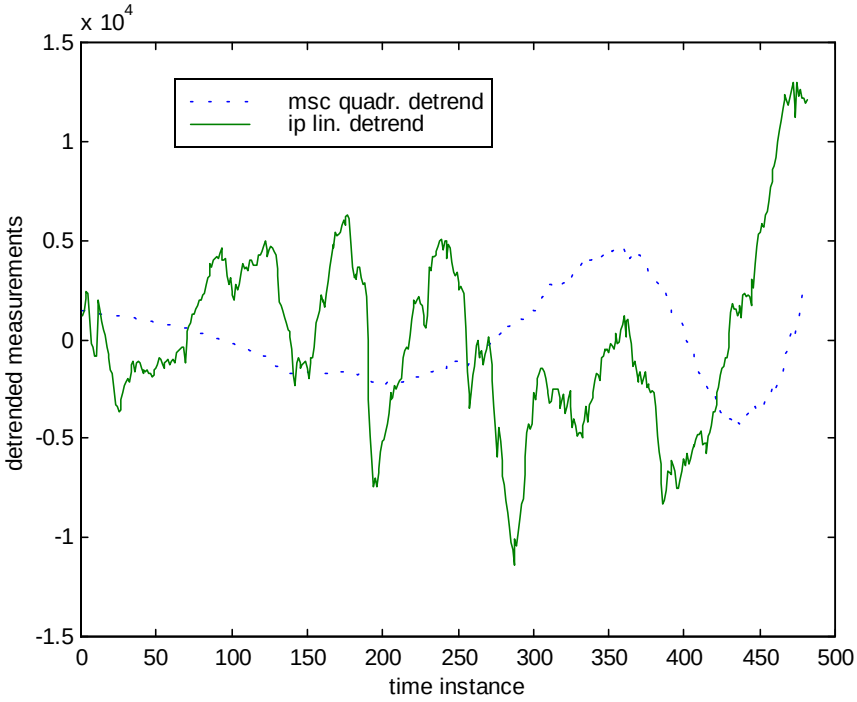


Figure 8.13 : Detrended curves

MSC was detrended by a quadratic curve and output IP by a linear curve, hence, giving the new 'detrended' variables msc2d and the output ip1d.

Figure 8.13 illustrates that a pattern recognition approach is doomed to fail, because the detrended output curve shows in its last part something that is not present in the first 2/3 of the curve, i.e. the training set.

A regression tree is grown from a maximal allowable mask with a memory depth of 60. The 1SE tree is depicted in Figure C.12. The forecasting results on the test set are depicted in Figure 8.14 (Figure C.13). They confirm the failure of the pattern recognition approach.

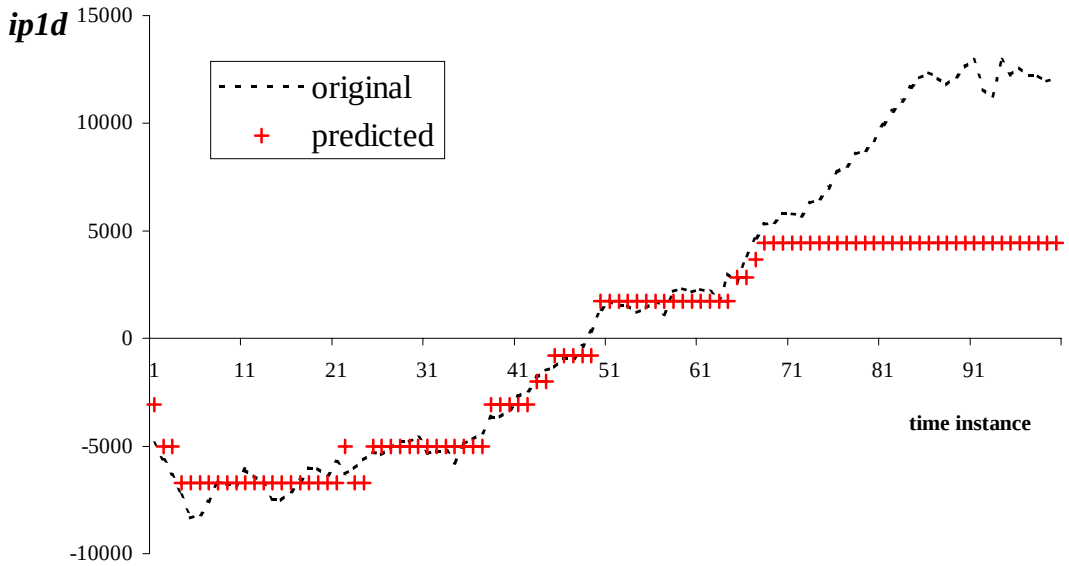


Figure 8.14 : Original and predicted output ip1d

Remarks:

- Maximal allowable masks with a memory depth of 60 for 4 variables is something unthinkable in SAPS-II. With the new data mining approach, such (entry) complex masks are now within reach. Consequently, systems that are more complex can be identified.
- Appendix C illustrated in addition that scatter plots of output versus *m*-inputs can not help much for detecting relevant patterns. Only if a splitter is near the top of the tree, one sees a pattern that is easily recognisable by regression trees.

8.6.3 The effect of missing values on forecasting

It is claimed in chapter 6 that an advantage of the data mining approach is that one can easily handle missing values. A kind of worst-case scenario is used in appendix C to show how regression trees perform in the presence of missing values. This worst-case scenario is based on the importance of variables (see Table C.1, Table C.2). Hence, as an experiment, quite some values from the variable ip1d(*i*-1) were removed directly from the static matrix of the test set. This is depicted in Figure C.15. If one realises how many times this variable is used for splitting (see Figure C.12), then a very bad forecasting may be expected at the corresponding time instances. Figure 8.15 shows that the forecasting is not as bad as may have been expected. Hence, the surrogate split performs quite well.

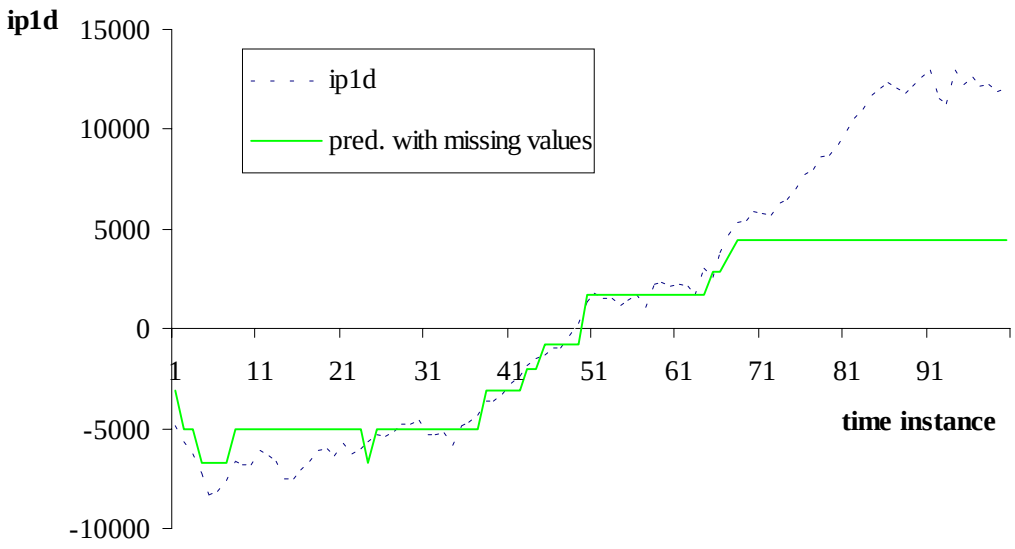


Figure 8.15 : Prediction with missing values

8.6.4 Feedback from tree classifiers to SAPS

Knowing which variables are important can give useful feedback to SAPS for constructing a candidate (primary) mask. Table 8.6 comes from appendix C (Table C.2). It shows the ranking of the variables according to their importance as primary splitters (no surrogate roles are considered). Additionally, Figure 8.16 (Figure C.12) shows where these variables are used in the tree.

Ranking	1(very high)	2 (very low, < 3 %)	3 (extremely low, < 1 %)
	ip1d(*i*-1)	ip1d(*i*-48)	ip1d(*i*-50), ip1d(*i*-18), msc2d(*i*-36)

Table 8.6 : Variable importance when only primary splitters are considered

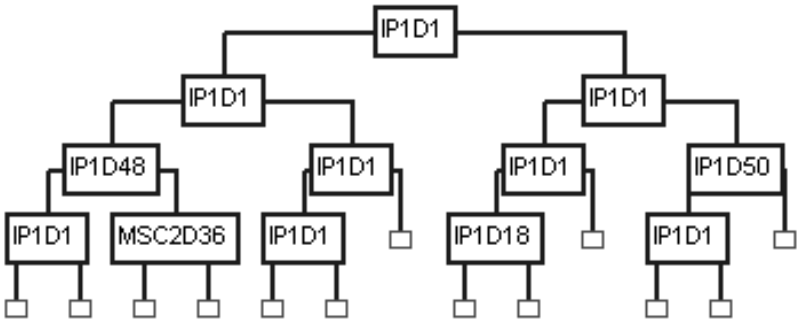


Figure 8.16 : The regression tree and its splitters

This information is fed back to SAPS in the form of a primary mask. The latter is given in Table 8.7 (Table C.4).

Reference index	CCP	FFR	msc2d	ip1d
-50	0	0	0	-1
-49	0	0	0	0
-48	0	0	0	-1
…				
-36	0	0	-1	0
…				
-18	0	0	0	-1
…				
-3	0	0	0	0
-2	0	0	0	0
-1	0	0	0	-1
0	0	0	0	1

Table 8.7 : Generating a mask via data mining

From this primary mask, an optimal search is done. The resulting optimal mask corresponds with the very simple relationship

$$\text{ip1d}(i) = \tilde{f}(\text{ip1d}(i-1))$$

The prediction with the 5UNN method based on the optimal mask is shown in Figure 8.17. It shows a good prediction. Additionally, Appendix C shows the forecasting with the 5+1NN method, which offers equally good forecasts.

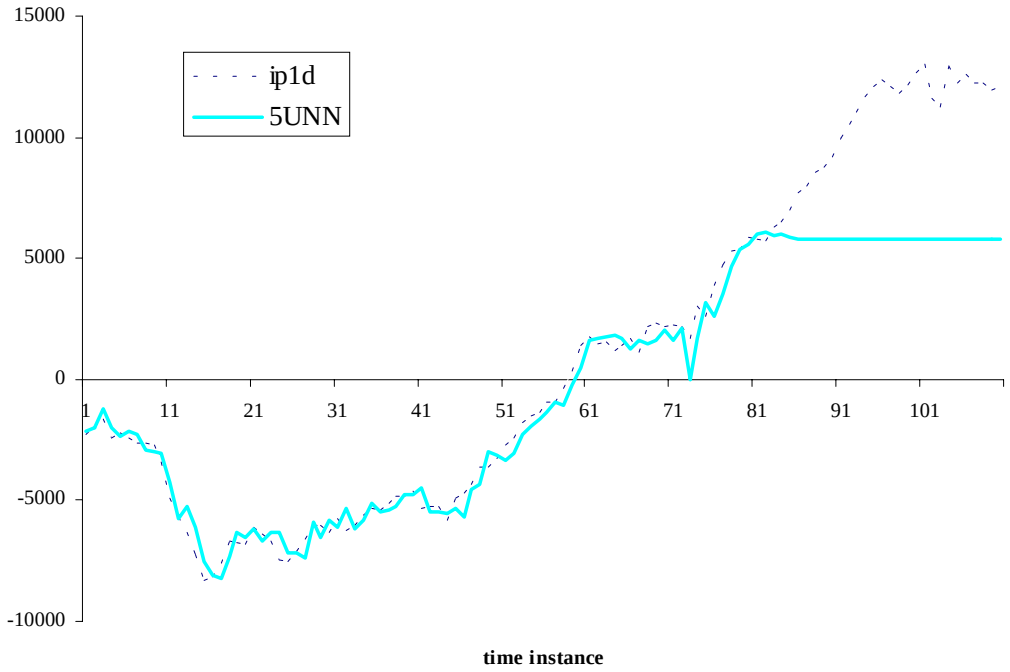


Figure 8.17 : Prediction 5UNN from optimal mask obtained via feedback

8.6.5 Looking at large data sets

Appendix D portrays data coming from a water distribution system in Portugal. The aim of this experiment is to try out how far one can go in the cardinality of maximal allowable masks. The example contains 15 input variables and 1 output. Each variable has 13128 observations.

A maximal allowable mask with memory depth 7 was created. It was also the computationally maximal allowable mask. Much deeper masks fell outside the specification of the CART version that was available (a 32MB version) [CART© 1999]. Hence, the option is taken to just show the principle on how one could try to extend a mask beyond the initial maximal allowable mask.

In inducing the tree, no cross-validation is used because the number of records is above the standard value (3000) for cross-validation. Hence, a (randomised) fraction of the data is used for testing. This fraction is set to 0.3333, so 2/3 is used for the learning set and 1/3 for the (internal) test set. From Table D.1 and Table D.2, it can be perceived that the variable n2 seems very important. The forecasting performance is good (Figure D.6).

A primary mask with a memory depth of at least 24 may give interesting results, because a pattern valid over 24h seems sensible for a water demand system (the measurements were taken hourly). To avoid violation of the computational limits of the CART implementation, a gap was created in the mask to reduce the cardinality. The earlier found pattern, which in-

cludes n2, is available in the new primary mask. In addition, a band of '-1' entries around a lag of 24h is created. A tree was grown from this primary mask. Except for some outliers, the forecasting seems quite good (Figure D.8). Both trees, which are grown in appendix D, give comparable forecasting results (Table D.6). Apparently, in the tree induction of the second tree the variable p24 is taken as first splitter instead of n2. This results in a slightly worse prediction and shows again that one should be careful towards induced models. A similar effect is present in the thesis of Nebot [1994].

8.7 Conclusion

This chapter demonstrates that the application of data-mining techniques, here embodied by regression trees, works. If one focuses on regression trees, then the rule-of-thumb approach towards recoding can be replaced by a dynamic quantisation that better exploits the information in the data. Even when one wants to do a-priori recoding, the KDD field shows two extra ways of doing that (in addition to the existing ones in SAPS). The ID3 algorithm is compared with the sub-optimal mask search. Although they behave alike (their functionality is quite similar), the former works in the measurement space and the latter in the model space (that is, it uses meta-modelling). Hence, refinement for the former can not be ported to the latter without careful consideration.

A major part of this chapter illustrates the benefits of applying regression tree to SAPS. Four examples, all of which are described in detail in the appendices, prove that the regression tree approach shows promising perspectives. Besides their benefits, they can give feed-back to the 'classical' SAPS tool by providing a primary mask of lower cardinality. A desirable by-product is that regression trees can automatically generate rule bases.

If comprehensibility is the major concern, regression trees can be simplified by pruning them back. As demonstrated, they still yield a reasonable good forecasting when compared with a sub-optimal search in SAPS.

A spin-off of the KDD framework is that two simplified and conceptually sound nearest neighbour methods are introduced, which work immediately on the 'flattened' raw data. The algorithms are very simple when compared to the original mixed nearest neighbour method from SAPS-II. Last but not least, the newly introduced algorithms perform equally well (sometimes better) with regard to forecasting accuracy.

From the two eager prediction methods, the state-observation matrix prediction method does not give accurate forecasts, but a regression trees can. In that aspect, the latter can stand up quite well against the lazy approach of the nearest neighbour methods.

References

CART © [1999], version 3.6 from Salford Systems, 1999 (www.salford.com).

Cellier F.E. [1991], Continuous System Modeling. Springer Verlag, New York, 1991.

Cellier F.E. and Yandell D.W. [1986], "*SAPS-II, Raw Data Analysis in CTRL-C*", User's manual and Progress Report, Department of Electrical and Computer Engineering, University of Arizona, Tucson, USA, 1986.

Cost S. and Salzberg S. [1993], "*A weighted nearest neighbor algorithm for learning with symbolic features*", Machine Learning, 10(1), p. 57-78, 1993.

Crombez J. [1999], personal communication, Universiteit Gent.

Kerber R. [1992], "*ChiMerge: Discretisation of Numeric Attributes*", Proceedings of the 9th National Conference on Artificial Intelligence, p. 123-128, MIT Press, July 1992.

Mathworks [1999], Matlab© version 5.3 from Mathworks (www.mathworks.com)

Mugica F. and Cellier F. [1993], "*A New Fuzzy Inferencing Method for Inductive Reasoning*", Proceedings of the Sixth Int. Symposium on Artificial Intelligence, and Intelligent Systems in Industry and Business, Monterrey, Mexico, p. 372-379, 1993.

Rulequest [1999], see www.rulequest.com

Siftware [1999], see www.kdnuggets.com

Van Welden D. [1998], Tree Classifiers as Data Mining Tools. MSc. Thesis, Catholic University of Leuven, Belgium, 1998.

Conclusion and Further Research

This thesis contributes to the induction of predictive models for black box systems, which has been inspired by GSPS and SAPS-II, in three major ways.
The first contribution is in the domain of general systems. It concerns the introduction of a sub-optimal mask search, which may give a new impetus to further research where masks that are more complex will be considered. In these it may be that derivatives may be included in the mask, more variables can be set forward (data augmentation), and larger time constants can be captured to find more valuable patterns. Not only has the new approach been theoretically justified, a prototype tool that supports the new search strategy has been implemented as well. It is intended as a research tool that can easily be modified and extended: it permits different recoding techniques to be combined, it implements different search methods, and it allows a diversity of quality functions to be dynamically constructed. The much-increased flexibility, however, comes at the cost of computing performance, which has proved to be a drawback when using masks with a very high cardinality.
The contributions in part two of the thesis transcend the boundaries of the GSPS framework: a bridge is laid from GST to the KDD. For this to be possible, a transformation of dynamic data to static data by means of a maximal allowable mask, results in a high dimensional data set, ripe for data mining. A wealth of available data mining methods can now be applied for subsequent data exploration. Chapter 5 and 6 briefly run through some of them. However, this plethora of methods can not be all explored. Therefore, this thesis concentrates on tree-classifiers (in particular regression trees) to illustrate the data mining approach to SAPS[1]. The benefits, which are plenty, are described in chapter 7 and 8. Surprisingly, regression trees, which do eager learning, perform very well in their forecasting when compared with the nearest neighbour methods. They forecast distinctly better than the state-observation method, which is another eager prediction method in SAPS. Taking further into account the comprehensibility of the tree classifiers, they can compete with the (lazy) nearest neighbour methods, which require the total learning set for each prediction. In addition, tree classifiers are at least equally worthy as masks, in their role as feature selectors for subsequent nearest neighbour prediction methods.
Another contribution concerns a strong improvement of the established (5+1NN) nearest neighbour method in SAPS. The newly introduced nearest neighbour methods, i.e. 5(U)NN, are competitive, and often better with regard to the forecasting accuracy than the existing 5+1NN method of SAPS. Because of their straightforward, sound underlying principles and their simple implementation, they have more potential for faster predictions.
While the former contributions lead to practical implications, the thesis contributes also to the pure theoretical domain. The link with hidden Markov models allows a rigorous formalisation and a clear defining of the underlying SAPS paradigm (chapter 3). The comparison between

[1] From now on, no further distinction is needed between SAPS-II and SAPS-ST

GST (and GSPS in particular) and the KDD domain opens a new world of research from which both fields can benefit via cross-fertilisation (chapter 7). Regrettably, it is impossible in this thesis to explore all the new fascinating possibilities for further research that are outlined in chapter 5. Nonetheless, the illustration with regression trees and the new nearest neighbour methods should give quite a good idea of the promising research opportunities.

When to use what?

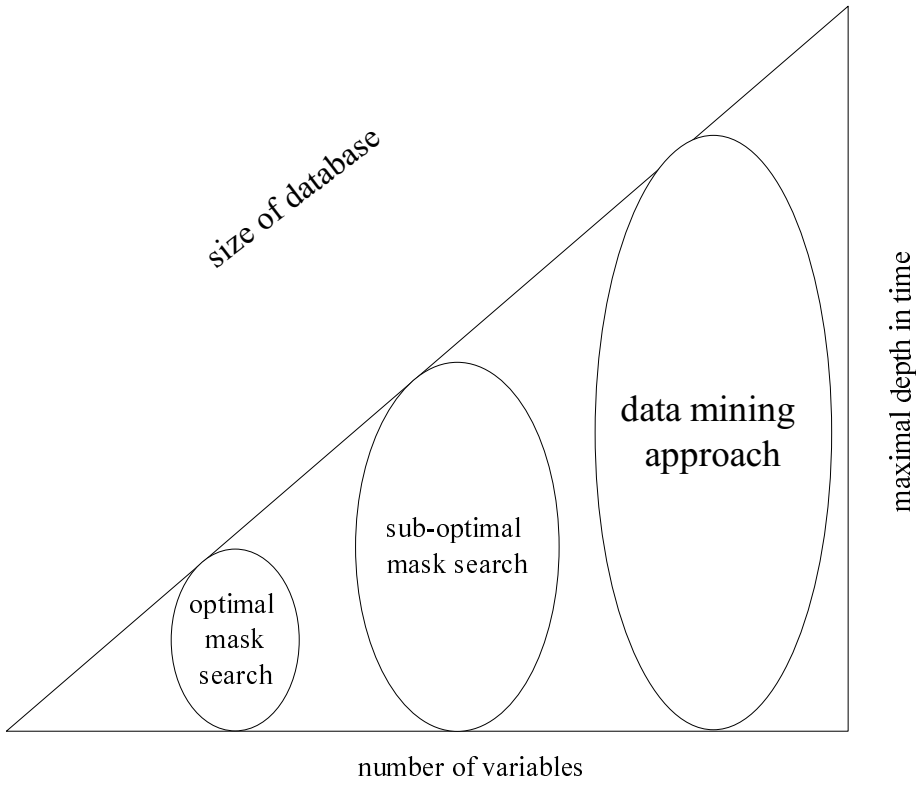


Figure 0.1 : Scope of techniques for different database sizes

At this point, three main methods to search for a predictive model are possible: the optimal mask search, the sub-optimal mask search, and the data mining approach (here represented by tree classifiers). They should be used in accordance with the data set size. The sub-optimal approach was invented to be able to tackle masks of higher cardinality, but the data mining approaches scale-up better. Figure 0.1 gives a suggestion of when to use which approach.

Further research

From the viewpoint of SAPS, the established link with the KDD domain results in some beneficial and useable concepts that are well known in this domain and that could be investigated further. Hence, the most obvious starting point for further research concerns the use of neural networks, genetic algorithms, and non-linear regression. Consequently, different model specifications can be tried. Figure 0.2 shows only some of them while giving an idea about the stepwise development of the research done in this thesis. For example, KDD includes unsupervised learning, which implies that clustering techniques may be utilised to find meaningful structured systems on epistemological level 3 of GSPS (structure level).
Surprisingly, appendix A showed that, on a test-set, fixed recoding can give more accurate forecasting results than uniform recoding. This is somewhat in contradiction with the fact that the latter retains more information (cf. chapter 2). Nevertheless, the dynamic discretisation done by tree classifiers appears to be better than these two recoding methods. However, this

effect is obscured because of the different validation paradigms used by SAPS and tree classifiers. The former relies on internal validation, while the latter uses external validation in its model induction. Therefore, an unbiased comparison is difficult without further research.

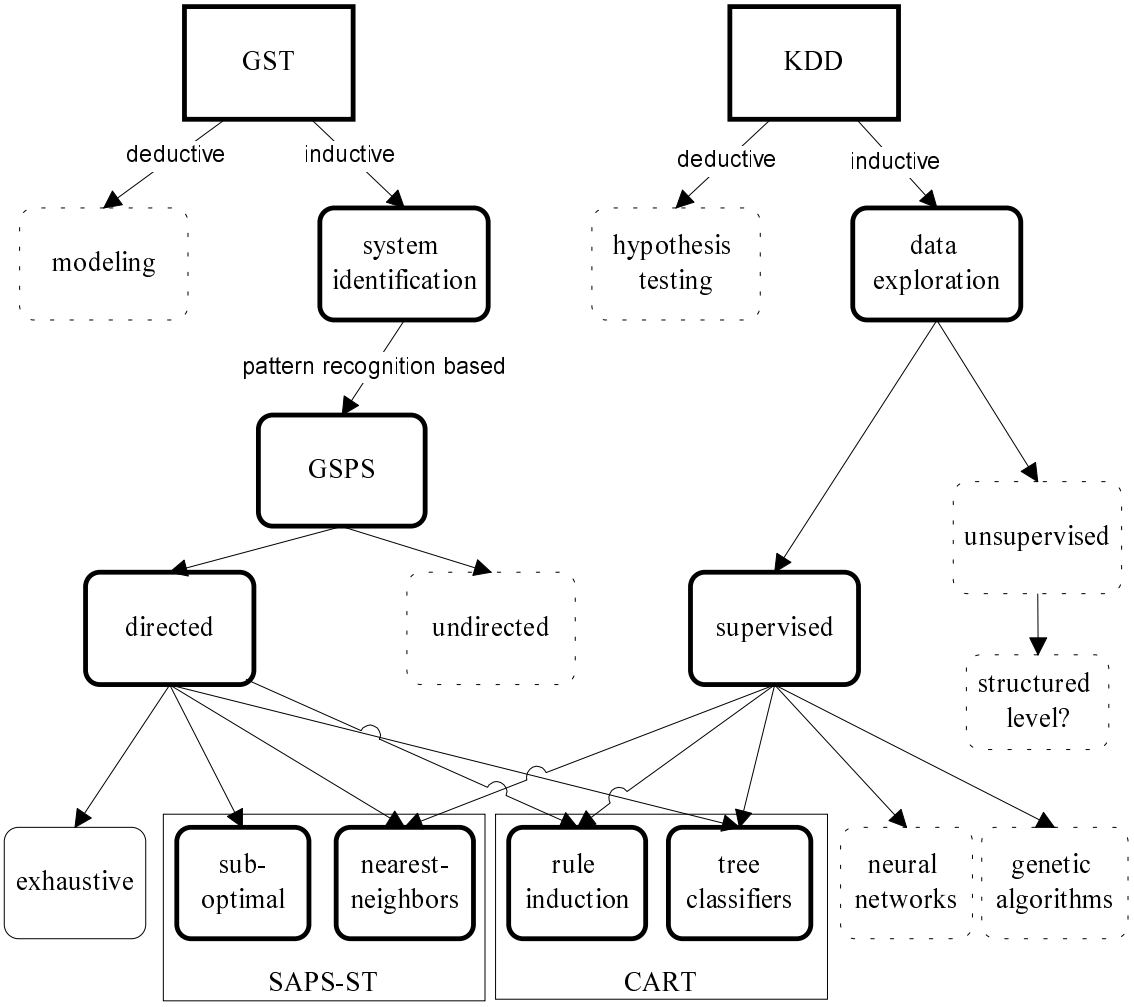


Figure 0.2 : The contributions, put in their context

Even when one stays with the tree classifier approach, additional research can be done. Though the examples in the appendices all had continuous output variables, this does not exclude the use of classification trees. For certain types of systems, some output ranges may be prohibited because they cause malfunctions of a component, or because they tend to destabilise the system itself. In that case, the output can be quantised in regions where some particular regions are more important than others. Consequently, the effect of a misclassification is different for these regions. Hence, an extension of error rate to cost matrices, risk matrices, or perhaps utility matrices can prove to be very beneficial.

APPENDICES

Prelude to the appendices

General issues that hold for all appendix examples

The data in every appendix is divided into a training set and a test set. The training set consists of 2/3 of the whole data set, while the test set contains the remaining 1/3[*]. Forecasting is always done on the test set.
If the quality of a mask is computed, it is always determined by the classical equation, given by (see chapter 2)

$$Q = H_r \times OR = \left(1 - \frac{H_m}{H_{\max}}\right) \times OR$$

For all predictions, the Sum of Squared Errors (SSE) and the Sum of Absolute Errors (SAE) is computed. When necessary, a plot of the predicted versus the original values for the test set is given to show the kind of deviations (Figure c). Such a plot always contains a 'diagonal' as reference line and sometimes the support. If all points lie on the diagonal, then there is a perfect forecasting.

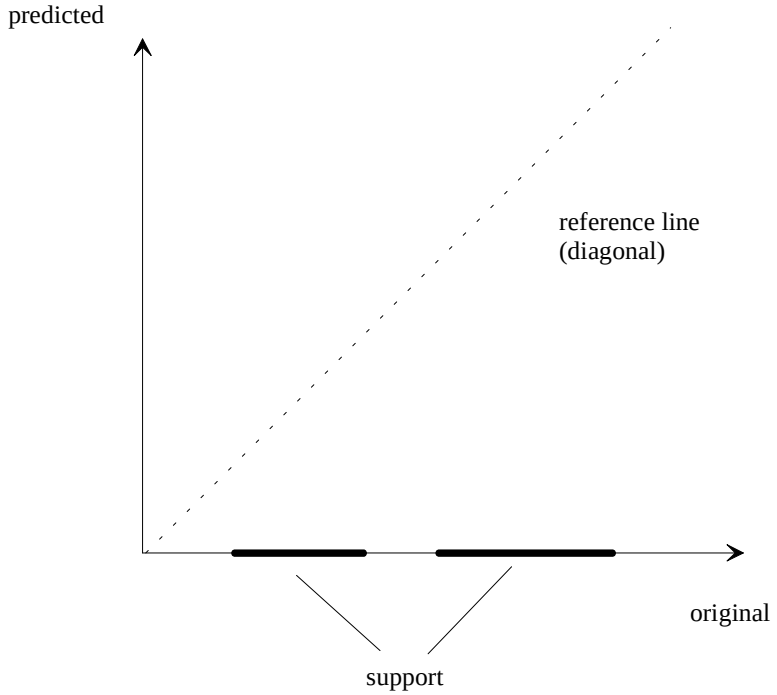


Figure c : General form of a 'predicted versus original' plot

In many examples, a multiple of forecasting methods is compared. The 'classical' SAPS-ST five nearest neighbour method is denoted by 5+1NN. The unit-rescaled five nearest neighbours method (based on raw data) is denoted by 5UNN. The simple non-unit-rescaled five nearest neighbours method (based on raw data) is denoted by 5NN.

[*] This is an often used rule of thumb. No investigation is done here to see the effect of other train-test partitionings.

Note about compatibility of SAPS-II and SAPS-ST

Van Welden [1999] established that SAPS-ST delivers the same results as SAPS-II in recoding, mask searching and forecasting, except for some small details. These are accounted for by transferring the recoding landmarks from SAPS-ST to SAPS-II. A perfect compatibility for mask searching can then be guaranteed. The forecasting delivers approximately the same results (within round-off errors). Consequently, it is justifiable to use SAPS-ST instead of SAPS-II for all comparisons made in the appendices.

Notations

Some output comes directly from the software package CART. Because of the restrictions it poses on variable names, the relative time indication is denoted in a shorthand manner. A variable *x* on time instance (*i*-*k*) will be abbreviated as *x_k*, e.g., var2(*i*-3) becomes var2_3.

Sources of data

The data in appendix A originates from the book of Cellier F., Continuous System Modeling, Springer Verlag, p. 586 - 593.

The data in appendix B is simulated via Simulink that accompanies Matlab (from Mathworks)

The data in appendix C is given by Crombez John, Universiteit Gent, Faculteit Economie. By this, the author likes to thank him for his contribution.

The data in appendix D is given by Cellier François. By this, the author likes to thank him for his contribution.

Appendix A

A Synthetic Linear System

A.1 Aim of the experiment

The aim of this experiment is

(a) to check if a different recoding results in a different optimal mask. If this is the case, one has an extra illustration of the subjectivity of an optimal mask, see sections A.3- A.3.2.

(b) to verify whether a newly introduced five nearest neighbours method forecasts on a test set as well as the 'classical' implementation of the 5+1NN method in SAPS, see sections A.3- A.3.2.

(c) to investigate if regression trees deliver equally good results as the 'classical' 5+1NN method from SAPS, see section A.4.

(d) to further compare the two eager methods: state-observation matrix forecasting and regression tree forecasting, see section A.4.1.

(e) to compare a simplified regression tree with other forecasting methods (especially the other eager method from (d), see section A.4.2.

(f) to demonstrate the automatic rule generation from within CART, see section A.4.3.

A.2 Description of data

Data is observed from a linear system that is described in [Cellier 1991]. The output variables are var2, var3 and var4, but only var2 will be considered as desired output (to forecast). Variables var3 and var4 are treated similar, but no relationship between var2, var3 and var4 on the same time instance is allowed. There are 201 observations in the training set and 100 observations in the test set (from 301 observations).

Inputs: var1: binary (0 or 1)

Outputs: var2, var3, and var4 are continuous variables, but only var2 will be considered here.

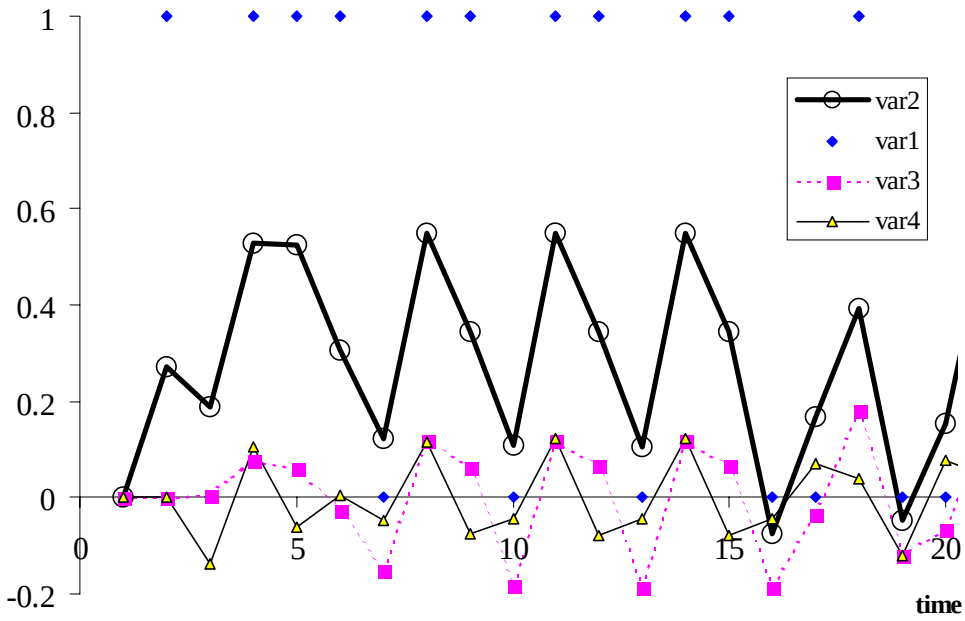


Figure A.1 : All signals (only part shown) as function of time

Figure A.2 shows all observations in the training set for output variable var2.

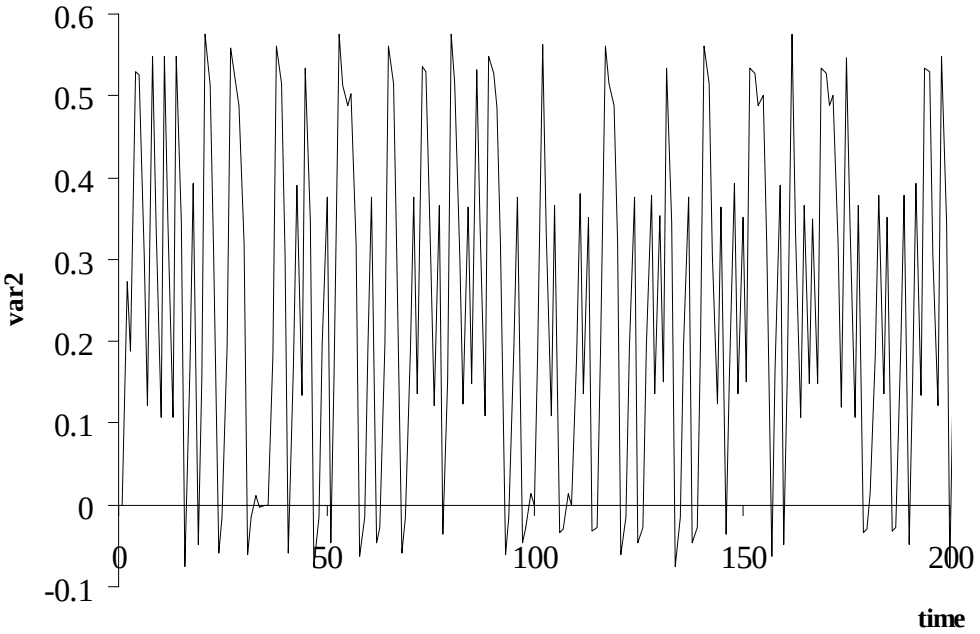


Figure A.2 : Plot of the response for the whole training set as function of time

A.3 Using an optimal mask search

A.3.1 Fixed recoding in SAPS-ST

Variable var1 is binary, which is recognised by SAPS-ST. Consequently, var1 is recoded automatically with rectangular membership functions for each interval. In this experiment, var2, var3 and var4 are recoded with a Gaussian membership function for each interval (see also chapter 4). The top of each interval corresponds with the respective quantitative values of the variable. Table A.1 shows the recoding specifications for variable var2. The leftmost col-

umn contains the set values of the interval. The second column gives the recoding function used (G = Gaussian, R = Rectangular, T = Triangular). The third column gives the top for the respective interval. The 'low Bound' column gives the lower bound, i.e., where the membership value should be zero for the neighbouring interval. A similar meaning is true for the 'high Bound'. 'landMark low' is the lower landmark, and 'landMark high' is the higher landmark for the interval in column 1. The recoding for var3 and var4 happens via the same algorithm. This is not shown here.

Interval	function	top	low Bound	high Bound	landMark low	landMark high
1	G	-0.076037	-0.076037	0.249977	-0.076037	0.141305
2	G	0.249977	0.032634	0.467319	0.141305	0.358648
3	G	0.57599	0.249977	0.57599	0.358648	0.57599

Table A.1 : Three-level fixed Gaussian recoding of var2

The primary mask put forward, is given by Table A.2. The mask search trajectory is depicted in Figure A.3. Remark that some peaks have approximately the same height.

var1	var2	var3	var4
-1	-1	-1	-1
-1	-1	-1	-1
-1	1	0	0

Table A.2 : Primary mask (maximum allowable mask) for search

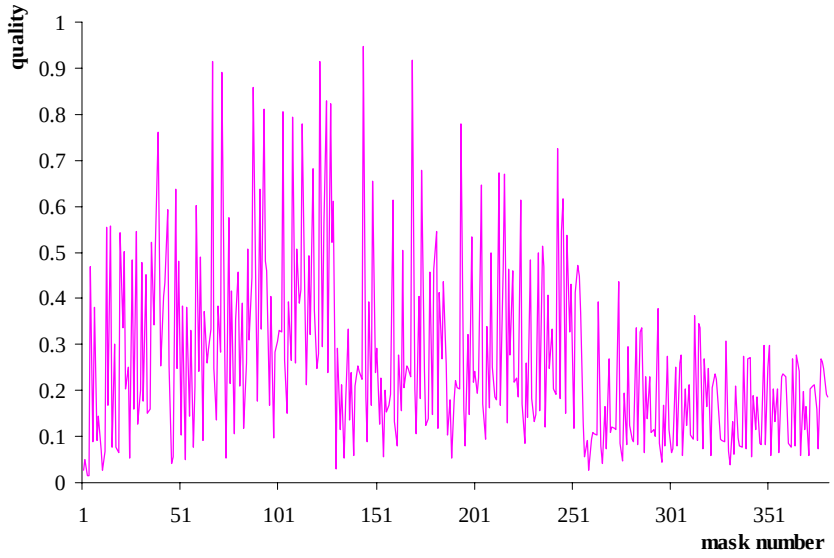


Figure A.3 : Quality trajectory in optimal mask search for fixed recoding

The optimal mask is found in Table A.3.

var1	var2	var3	var4
-1	-1	0	0
-1	0	0	0
-1	1	0	0

Table A.3 : Optimal mask under fixed recoding (quality = 0.9482)

The mask in Table A.3 stands for the qualitative dependency relation

$$\mathrm{var}\,2(i) = \tilde{f}\,(\mathrm{var}\,1(i-2), \mathrm{var}\,2(i-2), \mathrm{var}\,1(i-1), \mathrm{var}\,1(i))$$

Its corresponding state-observation matrix is listed in Table A.4. The input state vector components are found in column 1, 2, 3, and 4. Column 1 is var1(*i*-2), column 2 is var2(*i*-2), column 3 is var1(*i*-1), and column 4 is var1(*i*). Column 5 contains the probabilities of occurrence of the input state vector. The remaining columns are the observed output values. Each one is of the form 'set value; (recoding manner: membership value) (confidence & conditional probability)'.

2	3	2	1	(0.07)	-->	2;(G:0.842)	(11.24&1.0)	
2	1	1	1	(0.03)	-->	1;(G:0.975)	(3.37&1.0)	
2	2	2	2	(0.06)	-->	3;(G:0.966)	(7.8&1.0)	
1	3	1	1	(0.03)	-->	1;(G:0.968)	(3.28&1.0)	
1	2	1	2	(0.02)	-->	2;(G:0.680)	(2.53&1.0)	
2	3	1	1	(0.06)	-->	1;(G:1.0)	(11.28&1.0)	
1	2	2	1	(0.03)	-->	3;(G:0.525)	(3.13&1.0)	
2	2	1	2	(0.04)	-->	1;(G:0.522)	(3.63&0.88)	2;(G:-0.553) (0.55&0.13)
1	1	2	2	(0.05)	-->	3;(G:1)	(9.62&1.0)	
2	2	2	1	(0.02)	-->	2;(G:0.679)	(2.36&1.0)	
2	1	2	2	(0.02)	-->	3;(G:0.968)	(1.65&1.0)	
1	2	1	1	(0.05)	-->	1;(G:0.967)	(6.94&1.0)	
1	1	1	2	(0.08)	-->	2;(G:0.845)	(13.11&1.0)	
1	3	2	2	(0.02)	-->	3;(G:0.975)	(2.29&1.0)	
1	1	2	1	(0.08)	-->	3;(G:0.611)	(8.64&1.0)	
2	2	1	1	(0.04)	-->	1;(G:0.989)	(6.37&1.0)	
2	1	1	2	(0.03)	-->	2;(G:0.794)	(2.92&1.0)	
2	3	2	2	(0.05)	-->	3;(G:0.924)	(8.11&1.0)	
2	1	2	1	(0.03)	-->	2;(G:0.595)	(3.39&1.0)	
1	3	1	2	(0.03)	-->	2;(G:0.596)	(3.34&1.0)	
1	3	2	1	(0.03)	-->	2;(G:0.559)	(2.7&1.0)	
1	1	1	1	(0.05)	-->	1;(G:0.922)	(8.14&1.0)	
1	2	2	2	(0.05)	-->	3;(G:0.988)	(6.82&1.0)	
2	3	1	2	(0.06)	-->	1;(G:0.612)	(7.03&1.0)	

Table A.4 : State-observation matrix 1 under fixed recoding

Figure A.4 shows almost no difference between the original (test set) values and the forecasted values by the 5+1NN and the 5UNN method: forecasting is excellent in all cases.

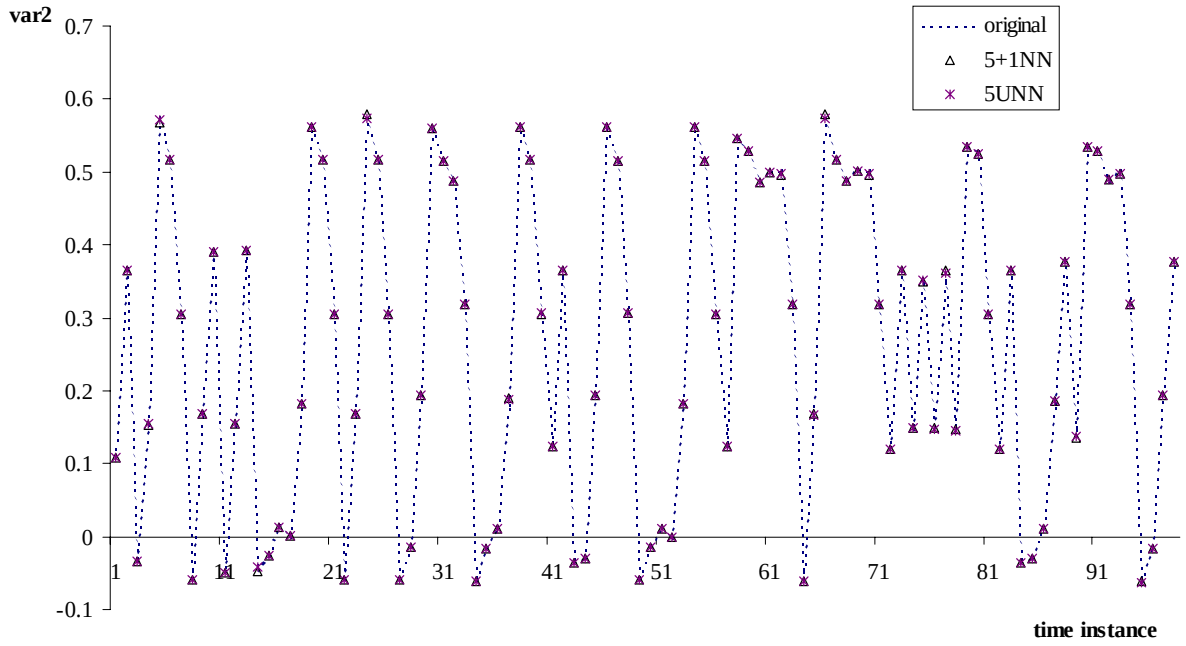


Figure A.4 : Forecasting results under fixed recoding

Figure A.5 zooms in on the differences between the original data and the forecasted ones. It shows that all methods perform equally well.

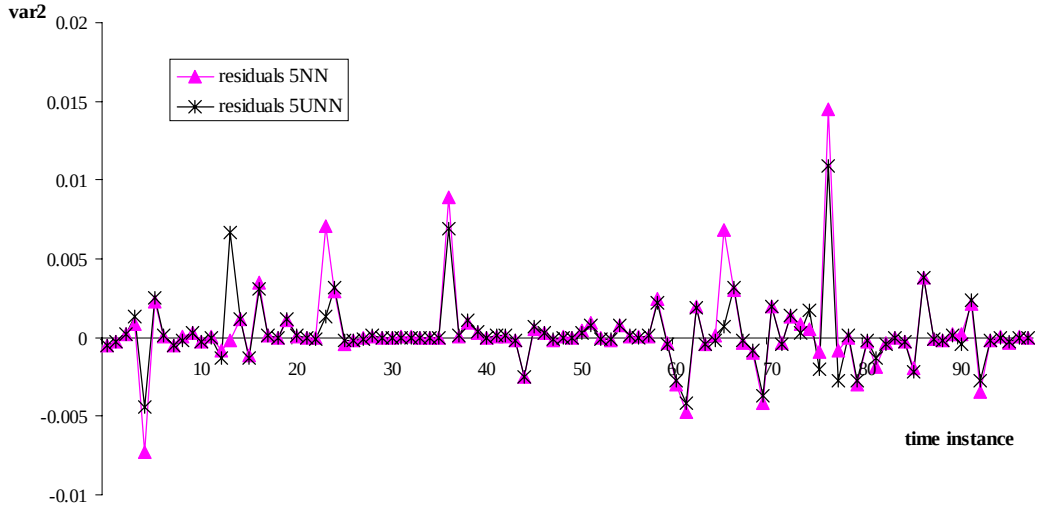


Figure A.5 : Zooming in on differences

On the average, the 5UNN forecasting method performs at least as well as the 5+1NN forecasting method. Table A.5 and Figure A.6 confirm these findings.

SSE		SAE	
5+1NN	5UNN	5+1NN	5UNN
0.000606	0.000398	0.117	0.105

Table A.5 : Looking at SSE and SAE for fixed recoding

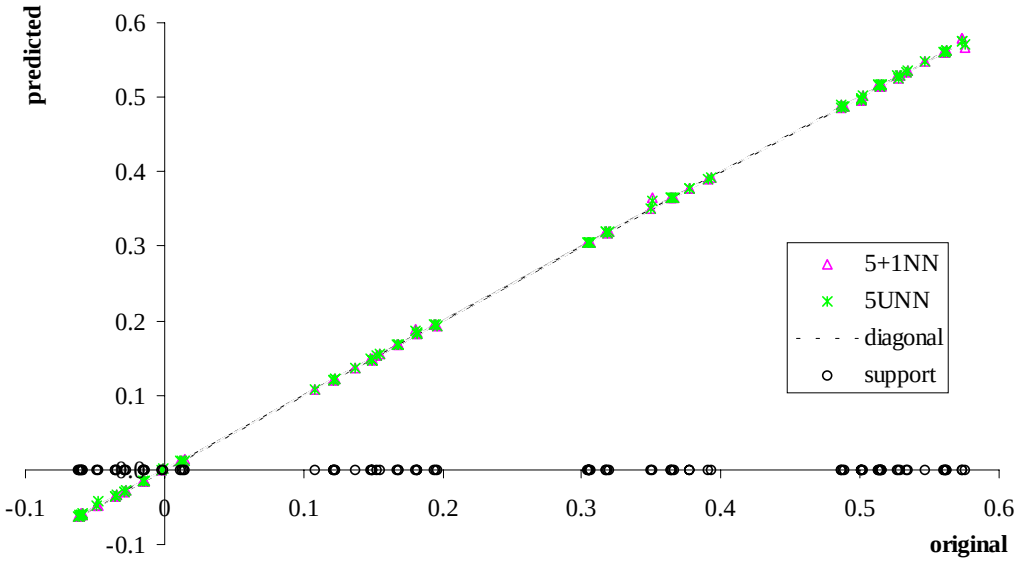


Figure A.6 : Plot of predicted versus original data for fixed recoding

A.3.2 Uniform recoding in SAPS-ST

Again, three intervals are taken for var2, var3 and var4. Variable var1 is recoded as in the fixed recoding experiment. Table A.6 displays the recoding results for variable var2.

Interval	function	top	low Bound	high Bound	landMark low	landMark high
1	G	-0.076037	-0.076037	0.249766	-0.076037	0.135128
2	G	0.249766	0.0295457	0.470197	0.135128	0.364404
3	G	0.57599	0.249766	0.57599	0.364404	0.57599

Table A.6 : Three-level uniform Gaussian recoding of var2

The primary mask is the same as in the fixed recoding case, but the quality trajectory for the optimal mask search, which is depicted in Figure A.7, differs.

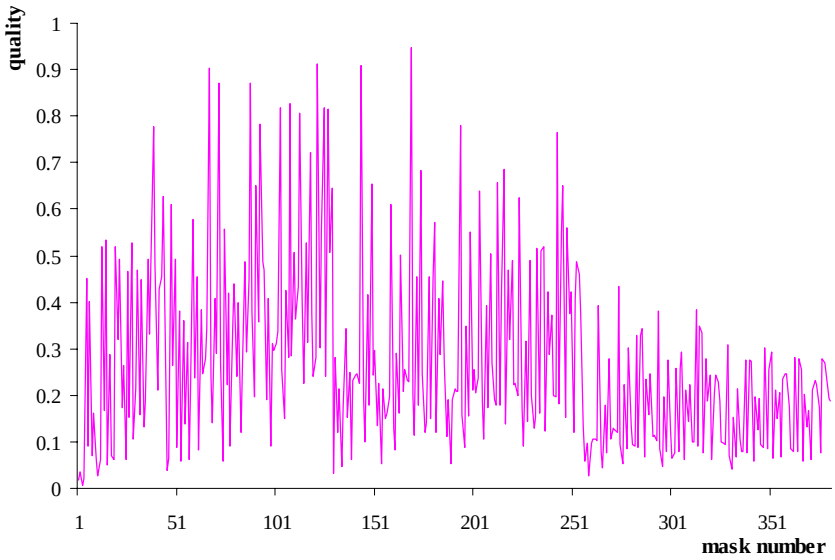


Figure A.7 : Quality trajectory in optimal search for uniform recoding

The optimal mask search results in optimal mask depicted in Table A.7.

var1	var2	var3	var4
-1	0	0	-1
-1	0	0	0
-1	1	0	0

Table A.7 : Optimal mask under uniform recoding (quality = 0.9464)

The mask in Table A.7 corresponds with the qualitative dependency relation

$$\text{var}\,2(i) = \tilde{f}\,(\text{var}\,1(i-2), \text{var}\,4(i-2), \text{var}\,1(i-1), \text{var}\,1(i))$$

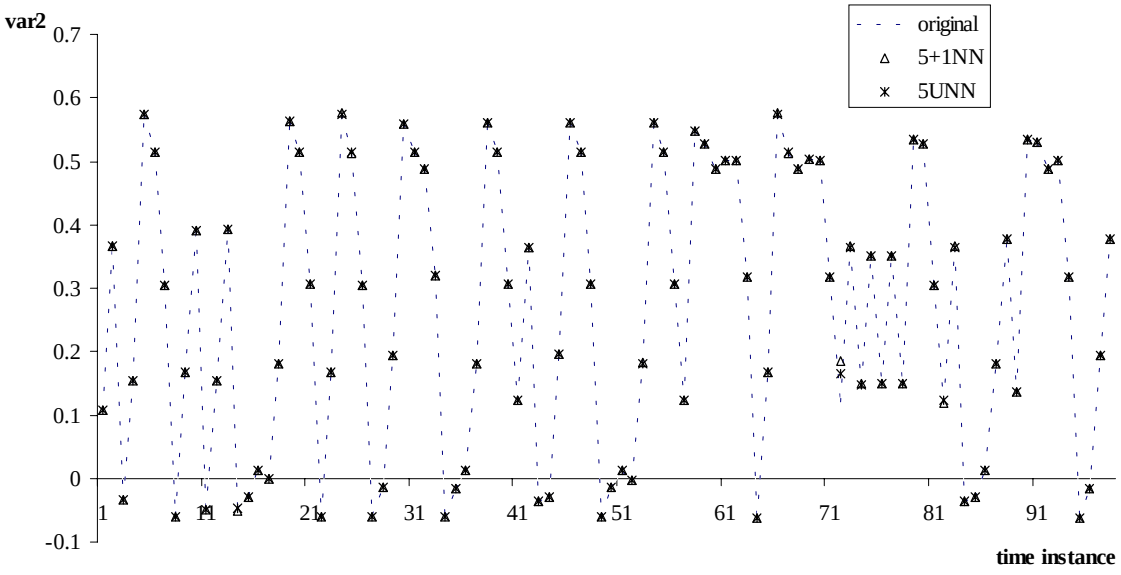


Figure A.8 : Forecasting results under uniform recoding

A comparison of forecasting of the 5+1NN method (see Figure A.8) with the 5UNN method shows again that the last is at least as good as the former one. Table A.8 and Figure A.9 confirm these findings.

SSE		SAE	
5+1NN	5UNN	5+1NN	5UNN
0.00416	0.00202	0.0991	0.0735

Table A.8 : Looking at SSE and SAE for uniform recoding

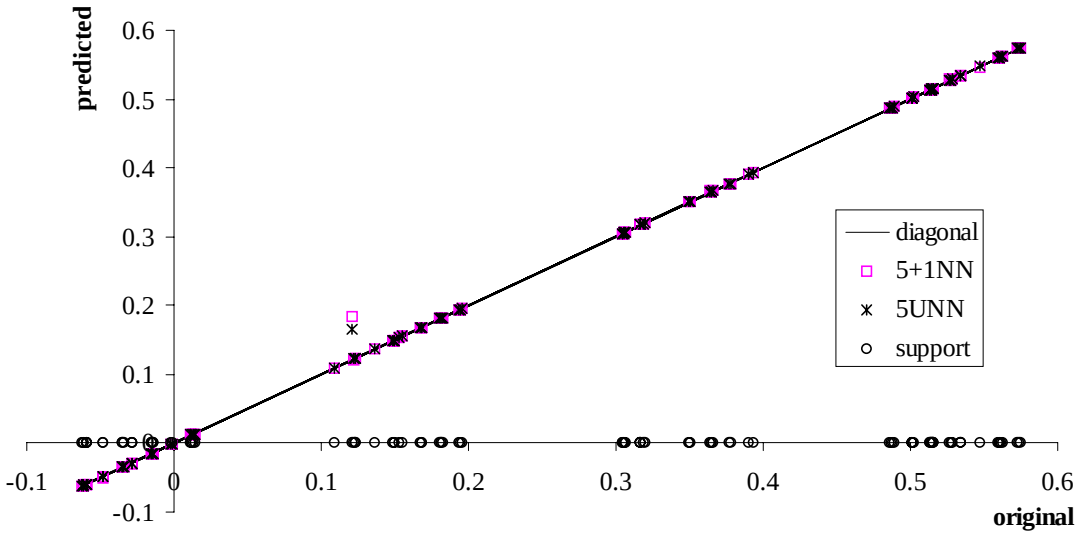


Figure A.9 : Plot of predicted versus original data for uniform recoding

Figure A.9 displays the predicted values versus the original values for the 5+1NN method and the 5UNN method. As for Figure A.6, the original values are also depicted on the x-axis to show why gaps occur in the plot.

Conclusion

One sees that the optimal mask under uniform recoding is different from the one under fixed recoding and vice versa. This experiment shows the influence of recoding (only fixed versus uniform is demonstrated here). Consequently, this confirms the justification for a sub-optimal approach. Obviously, when the number of intervals changes, one would also obtain different optimisation results.

When comparing the forecasting based on recoded (and regenerated) values or on unit-rescaled raw data, it is observed that the latter performs at least equally good (perhaps even a bit better).

A.4 Using regression trees

A.4.1 Use of the 1SE regression tree

The 1SE tree is selected, see chapter 6 and chapter 8. It has a cross-validated relative error of 0.05 %. Figure A.10 shows its position in the (cross-validated) relative error curve, together with the optimal and the simple, less than 1% error, tree.

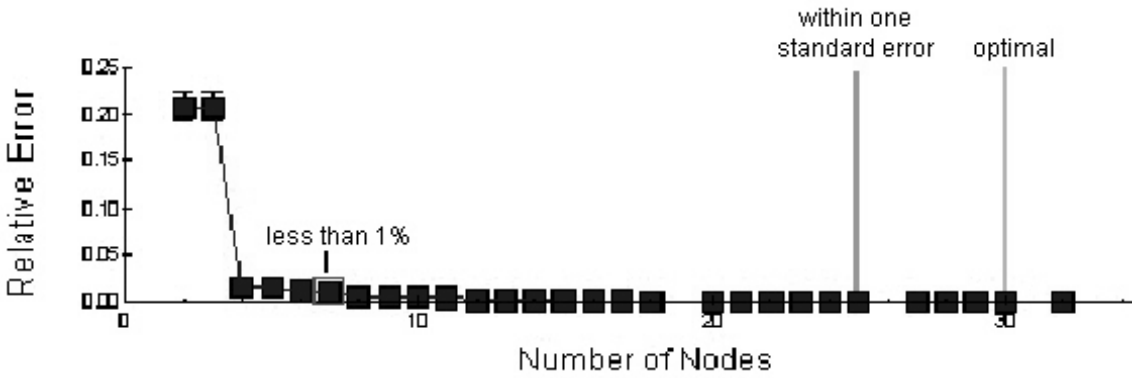


Figure A.10 : Relative cross-validation errors for different regression trees

The 1SE tree, which has 25 terminal nodes, is depicted in Figure A.11. The splitters are shown in the internal nodes.

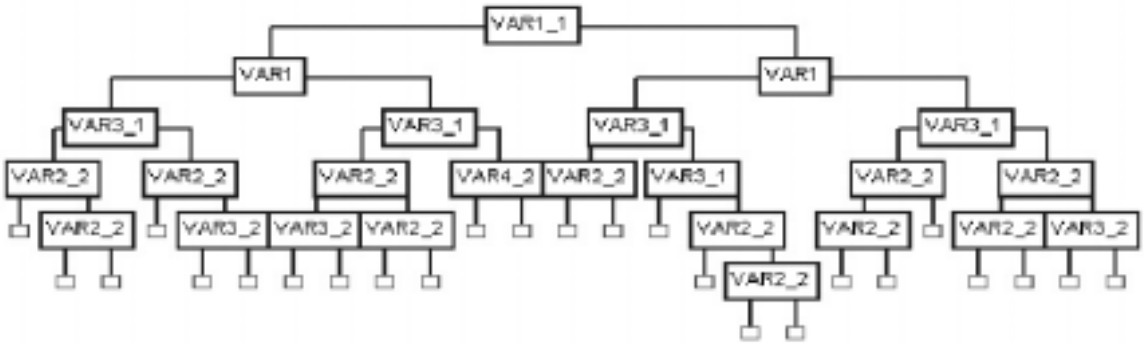


Figure A.11 : The 1SE regression tree with its splitters

The variability in the terminal nodes (denoted by the small squares in Figure A.11 is shown in Figure A.12 (notice the support set depicted in Figure A.6 or inspect Figure A.9 for an explanation of the gaps).

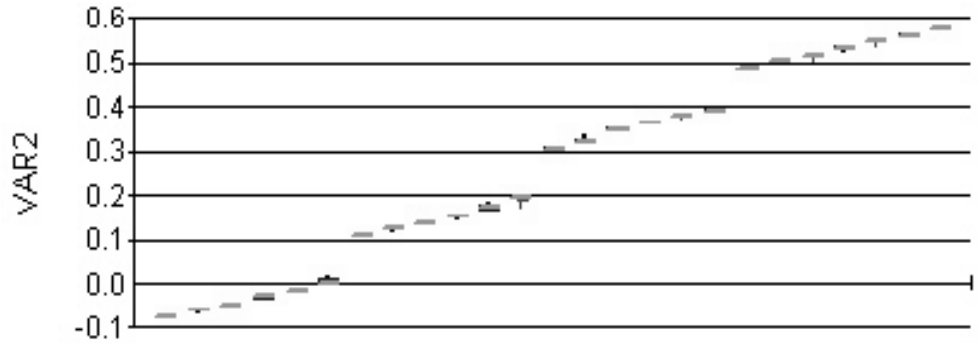


Figure A.12 : Box plots for terminal nodes and sorted by target variable prediction

A ranking of variables can be obtained from CART. The ranking can give feedback to SAPS, but this issue is postponed until appendix C. The ranking is shown in Table A.9. Remark that if surrogates are included, then no direct relationship can be detected with Figure A.11.

Ranking	1 (high)	2 (high)	3 (low)	4 (low)	5 (very low)
	var1(*i*-1)	var3(*i*-1)	var1	var4(*i*-1)	var2(*i*-2), …

Table A.9 : Ranking of most relevant variables (role as surrogates included)

If only the primary splitters are retained, var1(*i*-1) and var1 become high ranked (this can be seen from Figure A.11). This means that the others play a significant role as surrogates. The ranking gives a more pronounced description of the importance of variables compared to a mask.

Figure A.13 shows that the state-observation matrix method delivers the worst predictions. Despite its eager nature, the regression tree method performs equally well as the 5+1NN method (under fixed recoding).

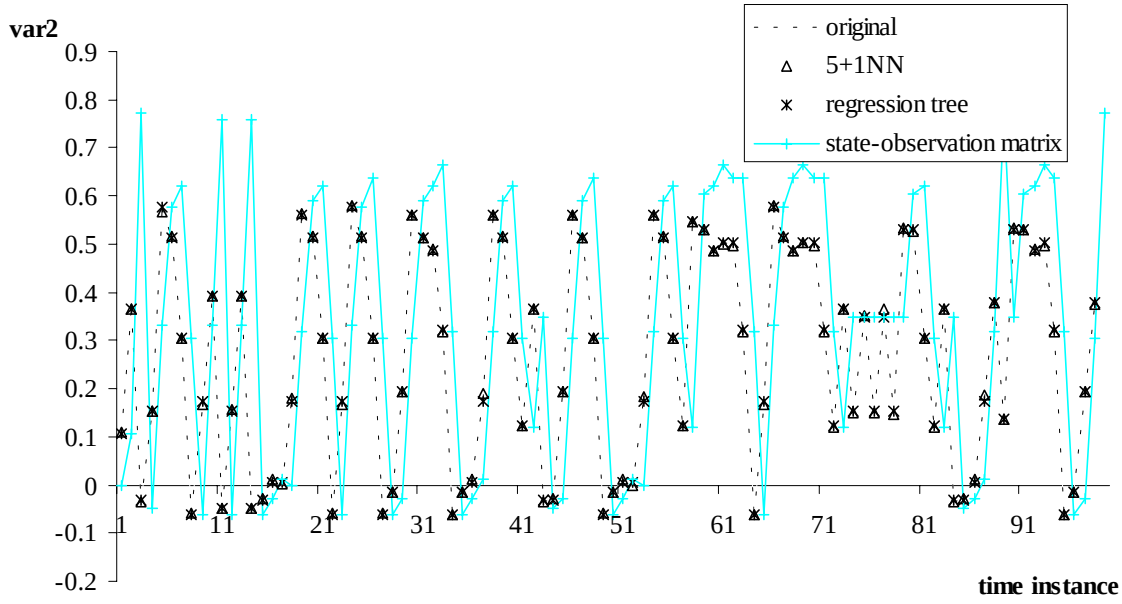


Figure A.13 : Forecasting with 5+1NN, regression trees and state-observation method

The differences between the two best forecasting methods are magnified largely in Figure A.14.

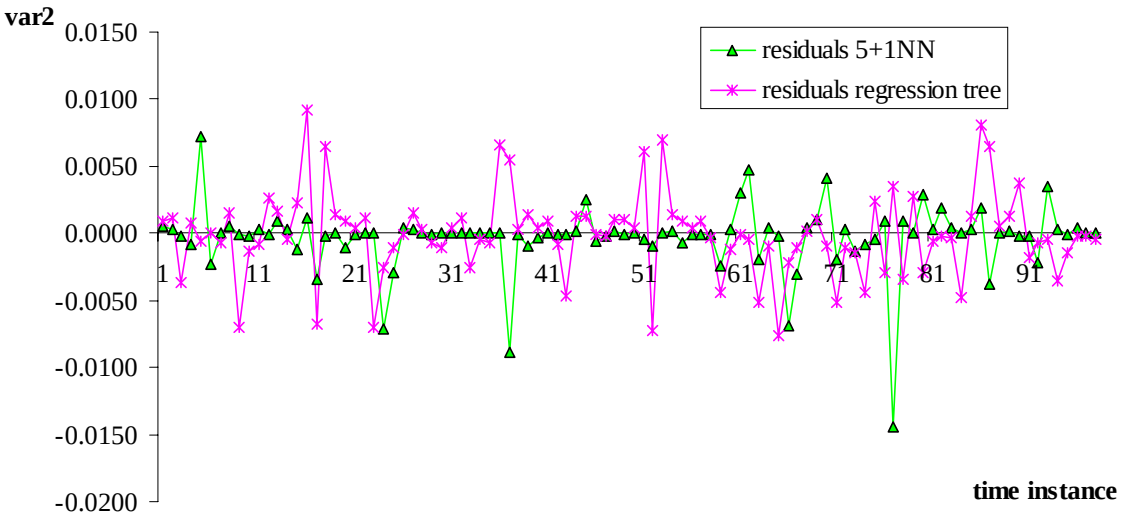


Figure A.14 : Zooming in on the 5+1NN method and the regression tree differences

SSE			SAE		
State-Observation	5+1NN, fixed recoding	Regression tree	State-Observation	5+1NN, fixed recoding	Regression tree
1.532450	0.000606	0.000962	7.601	0.117	0.213

Table A.10 : SSE and SAE for the three methods

Table A.10 and Figure A.15 confirm the conclusion drawn from Figure A.13 and Figure A.14. They show the good prediction performance of the 5+1NN method under fixed recoding and of the regression tree approach (the state-observation forecasting method was dropped because of its bad performance).

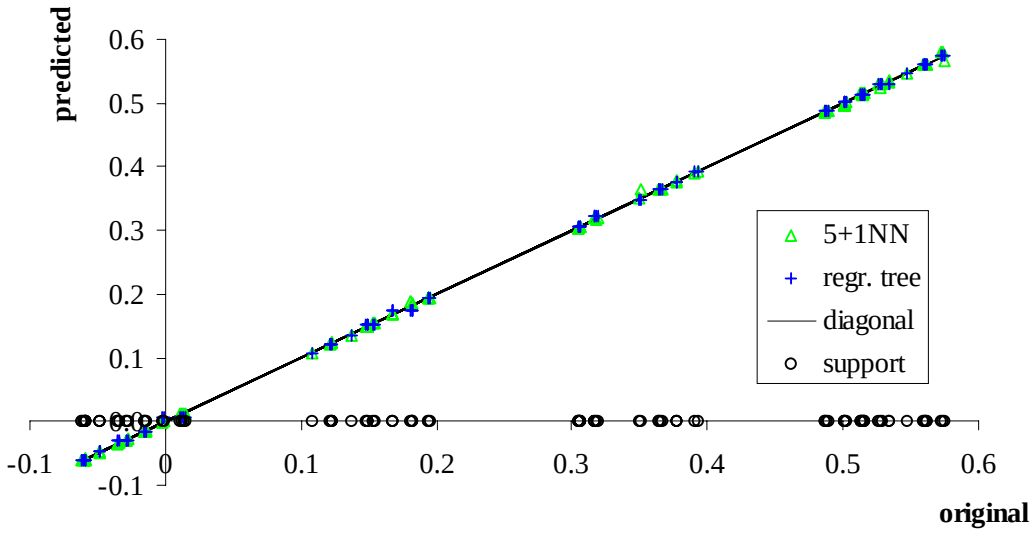


Figure A.15 : Plot of predicted versus original data for the regression tree approach

A.4.2 Use of a simple regression tree

One could also try an alternative tree with a relative cross-validation error of less than 1 %. The corresponding tree can be determined by Figure A.10. In this case, there are only seven terminal nodes. The tree structure, which shows the effect of further backward pruning of Figure A.11, is depicted in Figure A.16.

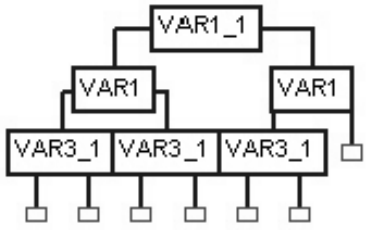


Figure A.16 : Smallest tree with a relative cross-validation error less than 1 %

The model is simpler and its accuracy is less. This can be observed from a comparison between Figure A.12 and Figure A.17.

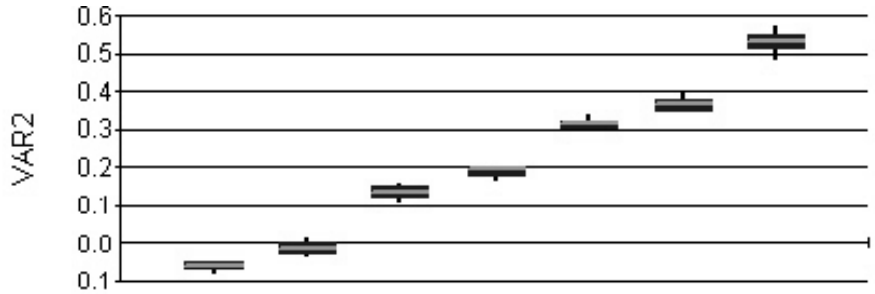


Figure A.17 : Box plots for terminal nodes sorted by target variable prediction for the less-than-one percent tree

The variable importance is the same as the previous tree, but var2(*i*-2) drops out because of the pruning.

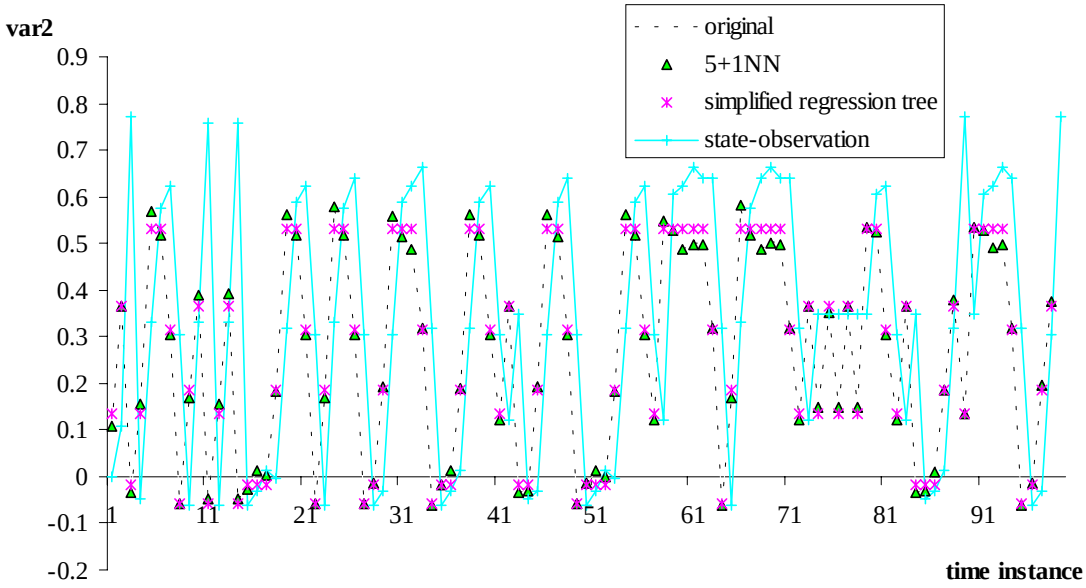


Figure A.18 : Forecasting with the 5+1NN, the regression tree and the state-observation method

Figure A.18 shows the forecast on the test set. It demonstrates that the state-observation matrix method again gives the worst predictions, the regression tree is much better, and the 5+1NN method gives the best predictions. This is not surprisingly, because the former two are eager methods and the latter is a lazy method. Despite its simplicity, the regression tree forecasting still outperforms the state-observation forecasting.

SSE			SAE		
State-Observation	5+1NN, fixed recoding	Regression tree	State-Observation	5+1NN, fixed recoding	Regression tree
1.532450	0.000606	0.0366620	7.601	0.117	1.486

Table A.11 : SSE and SAE for the usage of a simpler regression tree

Table A.11 confirms the conclusions drawn from Figure A.18. It is in concordance with the intuitive feeling that a simpler tree performs worse. It performs worse than the 5+1NN method, but this is not surprisingly because of the nature of the method (eager) and the parsimony of the regression tree.

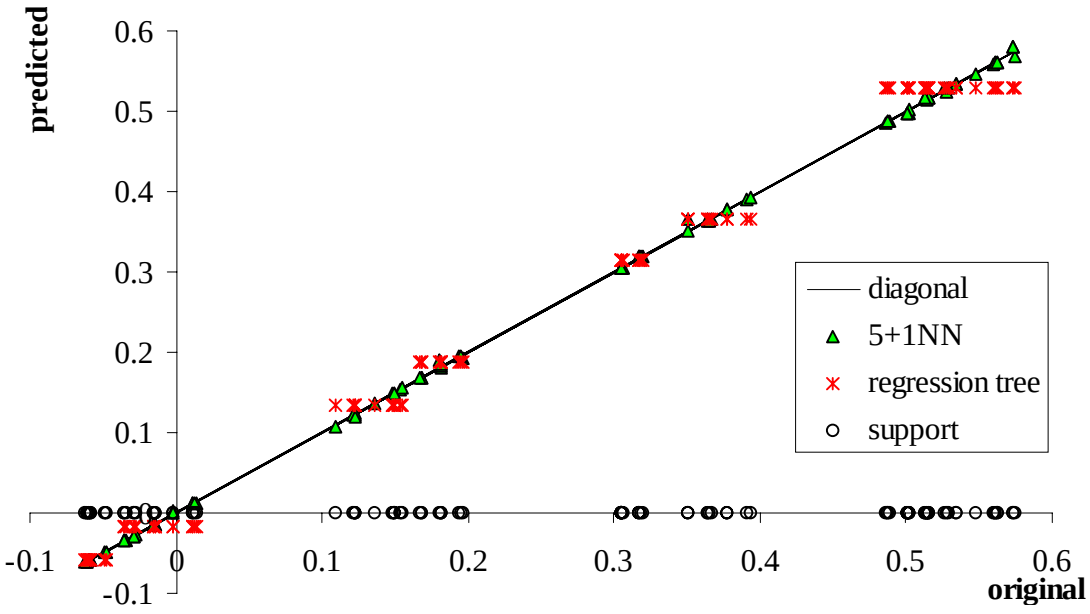


Figure A.19 : Plot of predicted versus original data for the simplified regression tree approach

Figure A.19 confirms that the simplified regression tree does not predict as well as the optimal regression tree and the 5+1NN method. This expected behaviour could also be explained from Figure A.17, which shows the larger variability in the terminal nodes around the relevant mean value.

A.4.3 Automatically generating rules from trees

CART can automatically generate rules for a given tree. Figure A.20 shows the output from CART for the tree in Figure A.16. These rules give implicitly a kind of quantisation of the relevant variables, tailored on the model.

```
/*Terminal Node 1*/
if
(
   VAR1_1 <= 0.500000 &&
   VAR1 <= 0.500000 &&
   VAR3_1 <= -0.102523
)
{
   terminalNode = -1;
   mean = -0.058293
}
/*Terminal Node 2*/
if
(
   VAR1_1 <= 0.500000 &&
   VAR1 <= 0.500000 &&
   VAR3_1 > -0.102523
)
{
   terminalNode = -2;
   mean = -0.016031
}
/*Terminal Node 3*/
if
(
   VAR1_1 <= 0.500000 &&
   VAR1 > 0.500000 &&
   VAR3_1 <= -0.050086
)
{
   terminalNode = -3;
   mean = 0.133416
}
/*Terminal Node 4*/
if
(
   VAR1_1 <= 0.500000 &&
   VAR1 > 0.500000 &&
   VAR3_1 > -0.050086
)
{
   terminalNode = -4;
   mean = 0.187156
}
/*Terminal Node 5*/
if
(
   VAR1_1 > 0.500000 &&
   VAR1 <= 0.500000 &&
   VAR3_1 <= 0.050068
)
{
   terminalNode = -5;
   mean = 0.313584
}
/*Terminal Node 6*/
if
(
   VAR1_1 > 0.500000 &&
   VAR1 <= 0.500000 &&
   VAR3_1 > 0.050068
)
{
   terminalNode = -6;
   mean = 0.365919
}
/*Terminal Node 7*/
if
(
   VAR1_1 > 0.500000 &&
   VAR1 > 0.500000
)
{
   terminalNode = -7;
   mean = 0.530348
}
```

Figure A.20 : Rule generation from CART

A.5 Conclusion

Table A.12 gives an overview of all prediction methods used in this appendix.

	SSE	SAE
State-observation forecasting	1.53245	7.601
5+1NN (fixed recoding)	0.00061	0.117
5UNN (fixed recoding)	0.00040	0.105
5+1NN (uniform recoding)	0.00416	0.099
5UNN (uniform recoding)	0.00202	0.074
Regression tree	0.00096	0.213
Simplified regression tree	0.03666	1.486

Table A.12 : Overview of all forecasting methods

Regression trees have good forecasting capabilities. In this experiment, they are comparable with the nearest neighbour methods with regard to their forecasting abilities. However, the former is eager, so it is preferable over the nearest neighbours if a fast classifier with a comprehensible structure is desired. A disadvantage is that a 'full' regression tree model is quite complex, but a nearest neighbour method does not provide insight at all. The regression tree is at least more comprehensible (with regard its model colour) than a nearest neighbour method. Even when simplifying trees, one still gets good predictions. Consequently, the use of regression trees for the purpose of prediction is promising. As eager method, it is also fast in forecasting.

If one intends to use a nearest neighbour method, then it is better to utilise one that is based on the raw (unit-rescaled) data (5UNN method) instead of on recoded triples. The latter gives theoretical complications that are not present in the former.

Surprisingly, although uniform recoding appears to be better from an information point of view (with regard to recoding), the fixed recoding gave (on the average) better forecasting results on the test set.

Eventually, the subjectivity of an optimal mask is demonstrated.

Appendix B

A Switched Noise-contaminated Sine System

B.1 Aim of the experiment

For this example, it is still possible to use both the optimal and the sub-optimal mask search. Hence, comparisons in which the two are relevant, can and will be done.

The purpose of this experiment is

(a) to demonstrate that even the second best optimal mask, which has a lesser quality, performs better on a test set than the optimal mask, see section B.3.1.

(b) to verify if the forecasting with the state-observation matrix related to other forecasting methods is really bad as seen in Appendix A, see section B.3.2.

(c) to see how the forecasting with the newly introduced 5UNN method compares with the existing 5+1NN method for the sub-optimal mask, see section B.3.2.

(d) to see how regression trees perform with regard to the other forecasting methods in SAPS, see section B.4.

B.2 Set-up and data generation

Data is generated from a sine wave (amplitude 10, frequency 1 rad/sec) on which random noise (mean 0 and variance 1, see Figure B.2) is added. The resulting signal is delivered to a switch, which is controlled by a block pulse generator that periodically blocks the passing of the by noise contaminated sine.

Inputs:

1. sine, rand : continuous
2. pulse: binary (0 or 2)

Output:

1. out: a continuous variable, which is periodically blocked by the switching

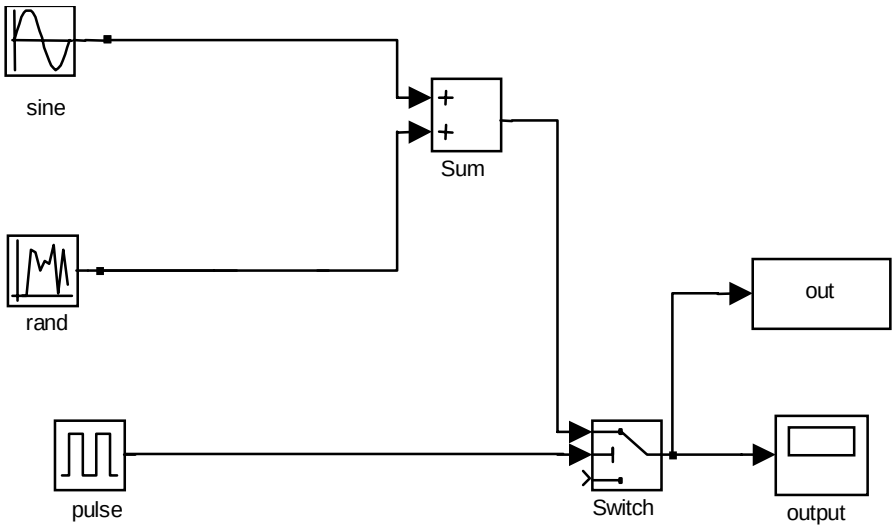


Figure B.1 : (Simplified) Simulink program that generated the data

The training set consists of 399 records. An excerpt from this data set (only the first 20 records are shown) is found in Table B.1, and a plot in Figure B.2. The test data contains 193 points.

Index	sine	rand	pulse	out	ref
1	9.093	0.0751	2	9.1681	
2	1.4112	0.3516	2	1.7628	
3	-7.568	-0.6965	2	-8.2645	
4	-9.5892	1.6961	0	0	
5	-2.7942	0.0591	0	0	
6	6.5699	1.7971	0	0	
7	9.8936	0.2641	0	0	
8	4.1212	0.8717	0	0	
9	-5.4402	-1.4462	2	-6.8864	
10	-9.9999	-0.7012	2	-10.7011	*i*-10
11	-5.3657	1.246	2	-4.1197	*i*-9
12	4.2017	-0.639	2	3.5627	*i*-8
13	9.9061	0.5774	2	10.4834	*i*-7
14	6.5029	-0.36	0	0	*i*-6
15	-2.879	-0.1356	0	0	*i*-5
16	-9.614	-1.3493	0	0	*i*-4
17	-7.5099	-1.2704	0	0	*i*-3
18	1.4988	0.9846	0	0	*i*-2
19	9.1295	-0.0449	2	9.0846	*i*-1
20	8.3666	-0.7989	2	7.5676	*i*

Table B.1 : First 20 records of raw data

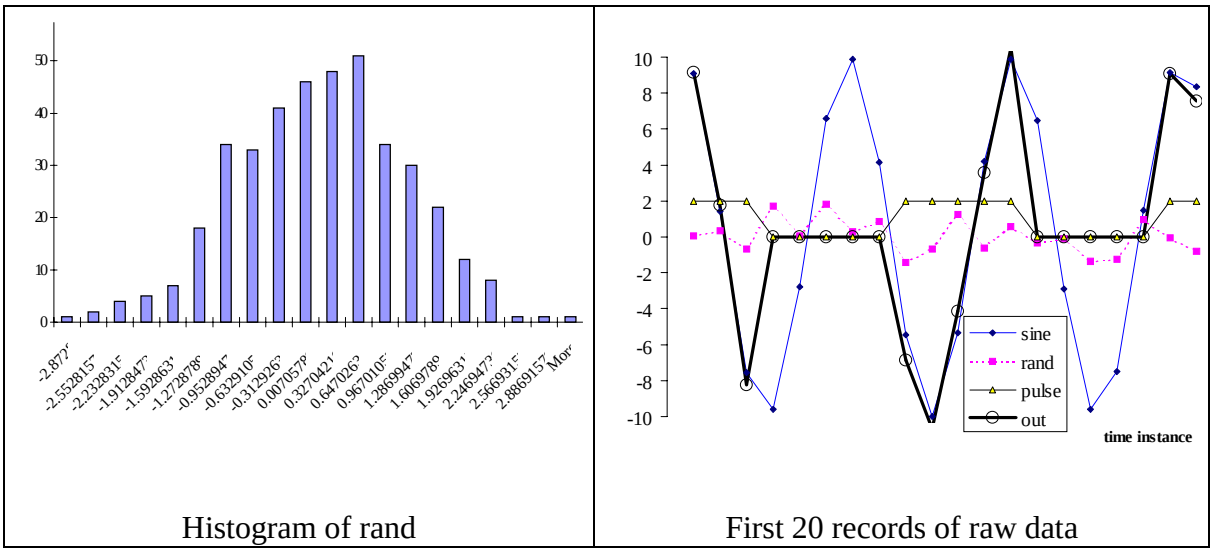


Figure B.2 : Description of data

A plot of the output for the whole training set is found in Figure B.3.

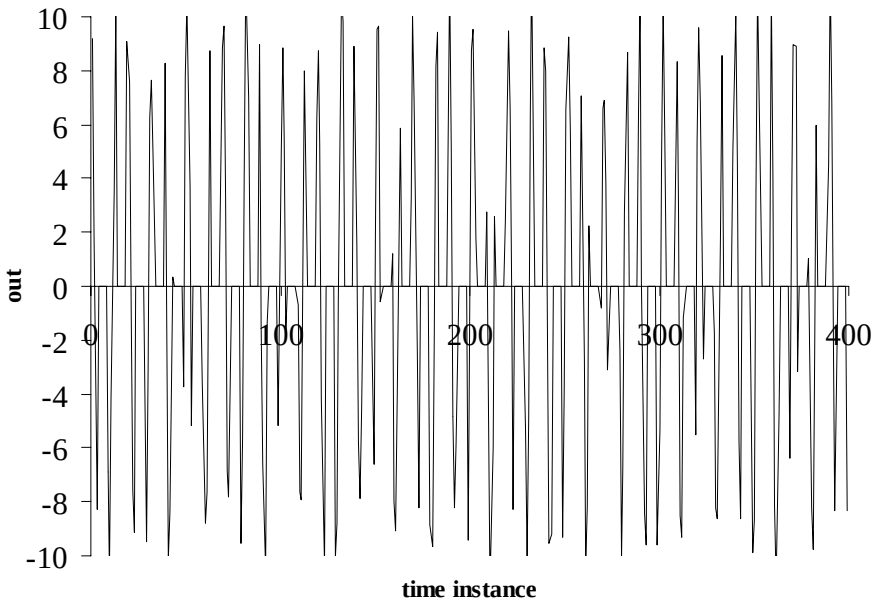


Figure B.3 : Plot of the response for the whole training set

B.3 Recoding and maximal allowable mask setting

Gaussian uniform recoding is used with 3 intervals for each variable, except for the pulse variable, where the systems detects that there are only two levels and no recoding is necessary (rectangular membership function). The maximal allowable mask has a depth of 11.

From this mask, two things will be done:

1. To perform an optimal mask search with SAPS-II (SAPS-ST was too slow for this, see section B.3.1), and to investigate its forecasting performance.
2. Start a sub-optimal mask search in SAPS-ST (see section B.3.2), because this method still works for deep maximal allowable masks. Hence, it is more universally applicable.

B.3.1 Optimal mask search and its relative optimality

For the optimal mask search there are $2^{43} \cong 10^{13}$ possible sub-masks[1] and this was bearably feasible on a Pentium-II 350Mhz (128MB) computer (it had to run for one night under SAPS-II). The sub-optimal mask search needs to generate 946 sub-masks, which takes much less time (but took a long time on the prototype in SAPS-ST).

The optimal mask, which has a quality of 0.9387, corresponds to the qualitative function

$$out(i) = \tilde{f}(\text{out}(i-10), \text{sine}(i-9), \text{pulse}(i-7))$$

The second best optimal mask, which has a quality of 0.9373, corresponds with

$$out(i) = \tilde{f}(\text{out}(i-10), \text{sine}(i-9))$$

The quality difference is very small. Comparing the respective state-observation matrices, which are shown in Table B.2 and Table B.3, reveals that for the second best optimal mask the state-observation matrix is most parsimonious. Furthermore, all combinations of the inputs are represented, so its predictiveness is 1.

1	1	1	(0.08)	-->	3;(G:1)	(23.36&1.0)		
1	1	2	(0.06)	-->	3;(G:0.999506)	(16.74&1.0)		
1	2	1	(0.05)	-->	3;(G:0.976319)	(12.24&0.85)	1;(G:0.554048)	(1.62&0.15)
1	2	2	(0.03)	-->	3;(G:0.938695)	(6.5&0.83)	1;(G:0.590596)	(1.14&0.17)
1	3	1	(0.02)	-->	1;(G:0.853634)	(4.6&1.0)		
1	3	2	(0.02)	-->	1;(G:0.768323)	(3.59&1.0)		
2	1	1	(0.07)	-->	2;(G:0.5)	(13.5&1.0)		
2	1	2	(0.1)	-->	2;(G:0.5)	(19.0&1.0)		
2	2	1	(0.07)	-->	2;(G:0.5)	(13.0&1.0)		
2	2	2	(0.1)	-->	2;(G:0.5)	(20.0&1.0)		
2	3	1	(0.06)	-->	2;(G:0.5)	(12.5&1.0)		
2	3	2	(0.1)	-->	2;(G:0.5)	(19.5&1.0)		
3	1	1	(0.02)	-->	3;(G:0.770285)	(4.13&1.0)		
3	1	2	(0.01)	-->	3;(G:0.808437)	(2.99&1.0)		
3	2	1	(0.05)	-->	1;(G:0.953463)	(10.37&0.79)	3;(G:0.625206)	(2.11&0.21)
3	2	2	(0.03)	-->	1;(G:0.967086)	(8.66&0.92)	3;(G:-0.638205)	(0.52&0.08)
3	3	1	(0.08)	-->	1;(G:1.0)	(25.11&1.0)		
3	3	2	(0.05)	-->	1;(G:0.995151)	(14.86&1.0)		

Table B.2 : State-observation matrix of optimal mask

[1] Luckily, the search stopped earlier because all masks with 4 *m*-inputs gave a lower quality. Hence, 'only' approximately 137000 masks were computed (about 3 masks/sec were computed).

1	1	(0.13)	-->	3;(G:1)	(40.1&1.0)		
1	2	(0.08)	-->	3;(G:0.976319)	(18.74&0.84)	1;(G:0.590596)	(2.76&0.16)
1	3	(0.04)	-->	1;(G:0.853634)	(8.19&1.0)		
2	1	(0.17)	-->	2;(G:0.5)	(32.5&1.0)		
2	2	(0.17)	-->	2;(G:0.5)	(33.0&1.0)		
2	3	(0.16)	-->	2;(G:0.5)	(32.0&1.0)		
3	1	(0.03)	-->	3;(G:0.808437)	(7.12&1.0)		
3	2	(0.08)	-->	1;(G:0.967086)	(19.03&0.84)	3;(G:0.638205)	(2.63&0.16)
3	3	(0.13)	-->	1;(G:1.0)	(39.97&1.0)		

Table B.3 : State-observation matrix of second best optimal mask

Therefore, the second optimal mask may be slightly preferable over the optimal mask (which also has a predictiveness of 1). Further sub-masks, which gave much less quality values, are not listed anymore. To investigate this further, one may look at the forecasting performance of the optimal and 'second' optimal mask. The 5UNN method is used for both masks. Figure B.4 shows that both give good predictions.

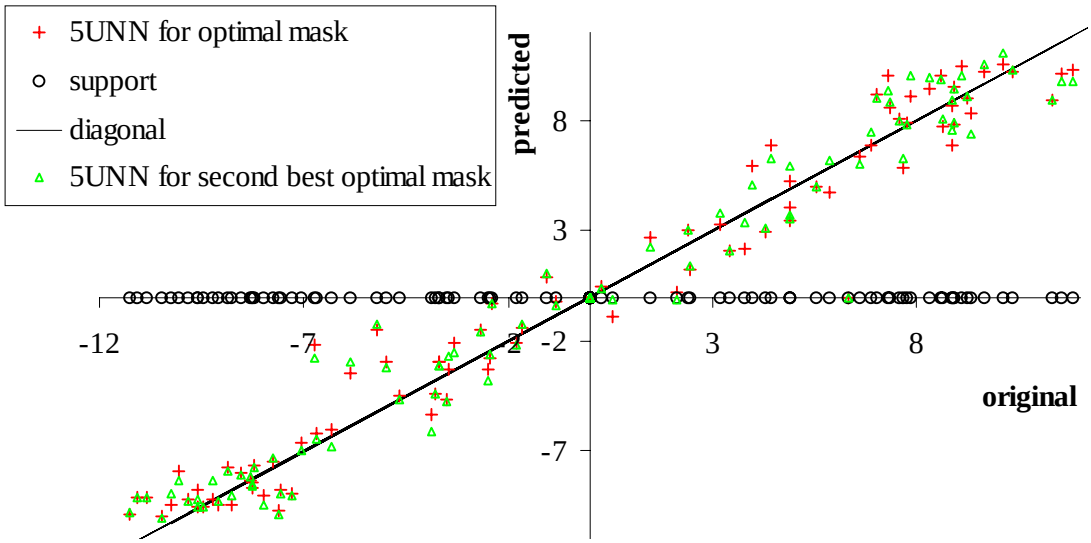


Figure B.4 : Predicted versus original output values for the 5UNN method under the optimal and second best optimal mask

The sum of squared errors and the sum of absolute errors for the optimal mask predictions is slightly less than the second best. The second best is simpler than the first, so it is preferable due to its greater simplicity (and less errors).

	SSE	SAE
Optimal mask	202.4	101.7
Second best mask	199.7	100.8

Table B.4 : SSE and SAE of optimal and second best mask

This again proves that *the optimal* makes no sense; the second best performs even slightly better in prediction on a test set.

B.3.2 Comparing forecasting under a sub-optimal mask search

The quality trajectory for the sub-optimal mask search is found in Figure B.5.

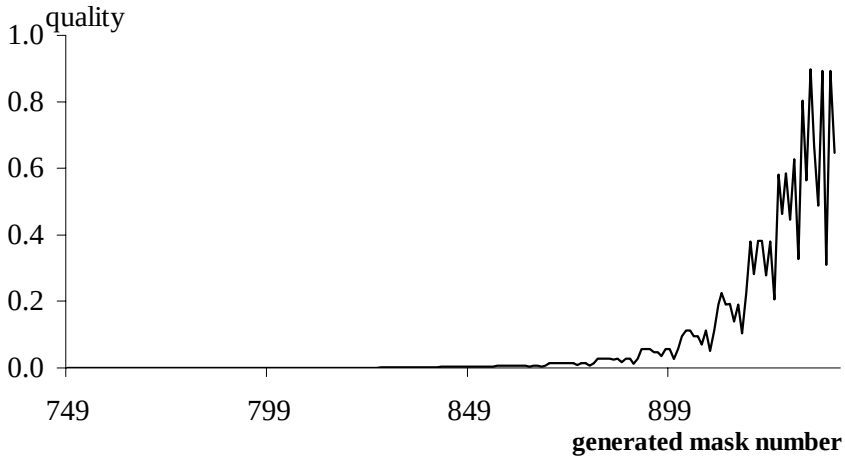


Figure B.5 : Trajectory of quality

The sub-optimal mask, which has a quality of 0.8973, corresponds with

$$y(i) = \tilde{f}(\text{sine}(i-9), \text{rand}(i-7), \text{pulse}(i-5), \text{pulse}(i-3))$$

Four forecasting methods are compared: the state-observation matrix method (Figure B.6), the 5+1NN method (Figure B.7), the 5NN method (Figure B.8), and the 5UNN method (Figure B.9 and Figure B.10).

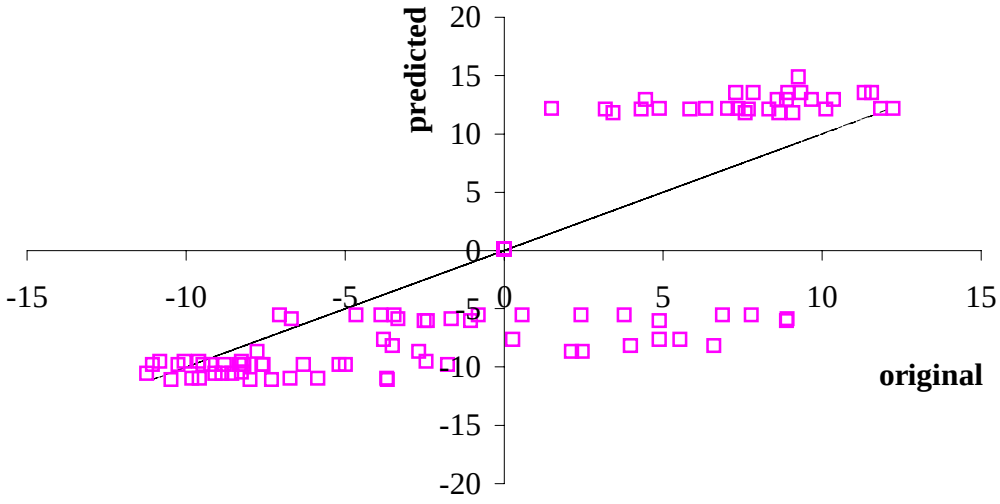


Figure B.6 : Predicted versus original output values for the state-observation matrix method under the sub-optimal mask (SSE = 3485, SAE = 454.9)

If a state could not be observed, the second, third, … best sub-optimal masks (and accompanying state-observation matrices) were used instead. They correspond with

- $y(i) = \tilde{f}(\text{sine}(i-9), \text{pulse}(i-5), \text{pulse}(i-3))$ (quality = 0.8923)
- $y(i) = \tilde{f}(\text{sine}(i-9), \text{rand}(i-7), \text{pulse}(i-5))$ (quality = 0.8920)
- …

Figure B.6 shows that the forecasting is not good at all.

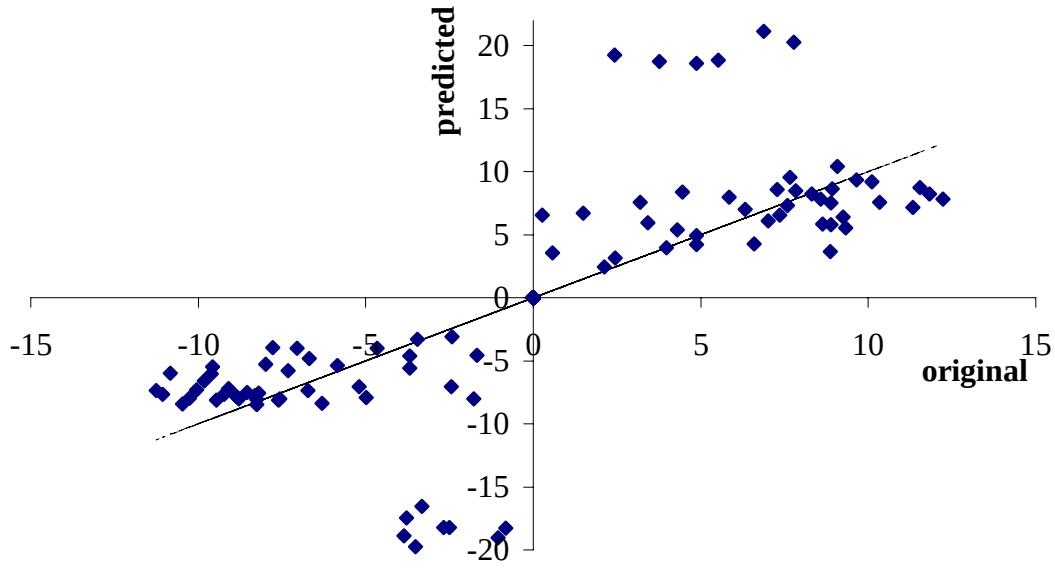


Figure B.7 : Predicted versus original output values for 5+1 NN method under the sub-optimal mask (SSE = 3723 and SAE = 369.4)

The forecasting in Figure B.7 is comparable with the state-observation-based forecasting in Figure B.6. The SSE is somewhat worse (larger) and the SAE better (lesser), which indicates that there are less deviations, but of a larger magnitude.

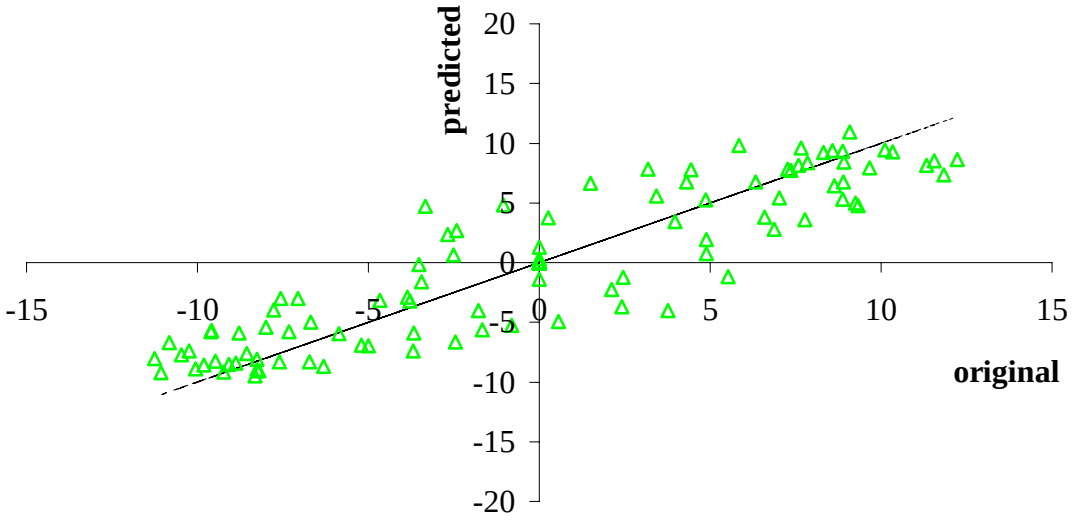


Figure B.8 : Predicted versus original output values for 5NN method under the sub-optimal mask (SSE = 932 and SAE = 242.9)

Forecasting with the 5NN method is much better than the previous methods. There are no large systematic deviations. Figure B.9 and Figure B.10 show the forecasting performance of the 5UNN method. It can be seen that the 5UNN method is comparable with the 5NN method. This is confirmed in Table B.5.

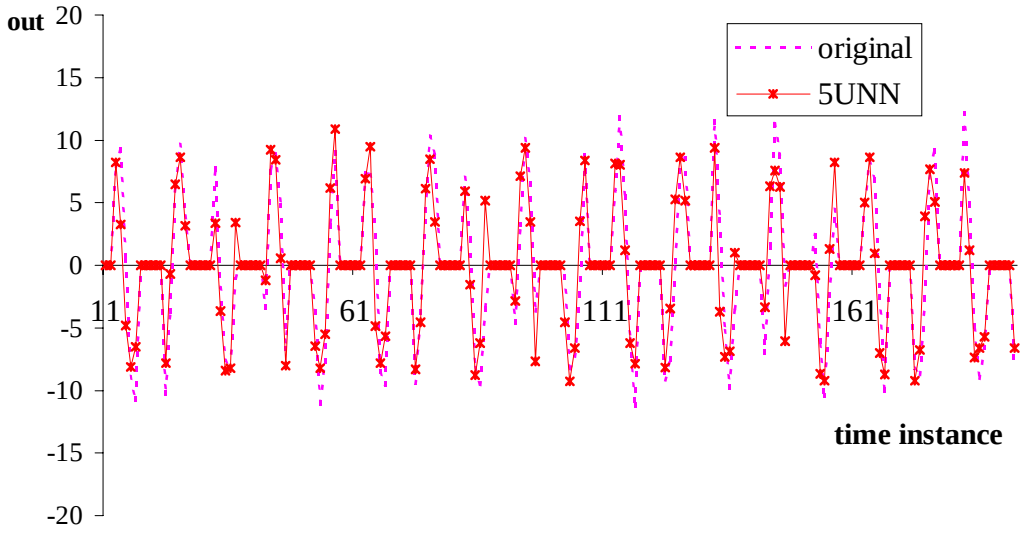


Figure B.9 : Predictions by the 5UNN method under the sub-optimal mask

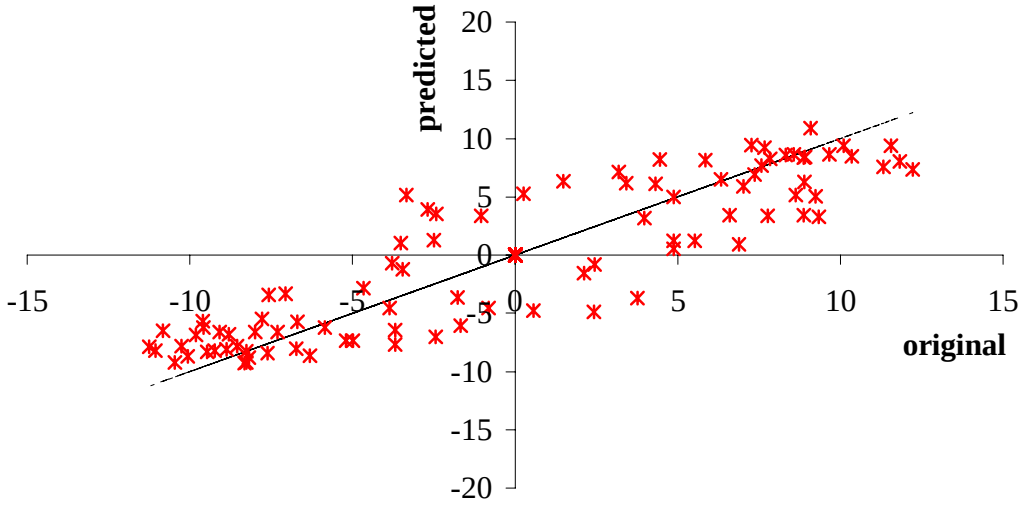


Figure B.10 : Predicted versus original output values for 5UNN method under the sub-optimal mask (SSE = 1011 and SAE = 249.3)

Conclusions for forecasting

	SSE	SAE
State-observation based	3485	454.9
5+1NN	3723	369.4
5NN	932	242.9
5UNN	1011	249.3

Table B.5 : Table of SSE and SAE under the sub-optimal approach

The forecasting methods based on the raw data are much better than the state-observation method and the 5+1NN method under the sub-optimal approach.

Comparison of Table B.4 and Table B.5 shows the obvious result that the optimal mask can do a better forecasting than the sub-optimal one.

B.4 Using a regression tree for identifying a model

The 'flattening' transformation, with a maximal allowable mask of depth 11, results in 389 'static' records for the learning set. CART is used to find the 1SE tree. The latter is a regression tree with 14 terminal nodes, which is depicted in Figure B.11.

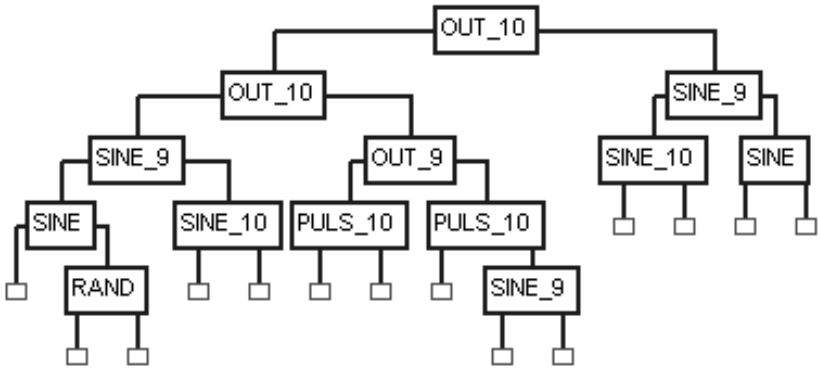


Figure B.11 : Selected tree in CART

The ranking of the variables in Figure B.11 is given by Table B.6.

Ranking	1 (very high)	2 (medium)	3 (very low)
	out(*i*-10)	sine(*i*-9)	sine, sine(*i*-10), pulse(*i*-10), out(*i*-9), rand

Table B.6 : Ranking of variables (only primary splitters)

Apparently, the output on 10 time steps before appears to play a crucial role, which is not surprising if one considers Table B.1. The sine plays a less, but still important role at a lag of 9 time steps. The pulse signal plays a lesser role, and the random variable has no real influence. This is consistent with Figure B.11.

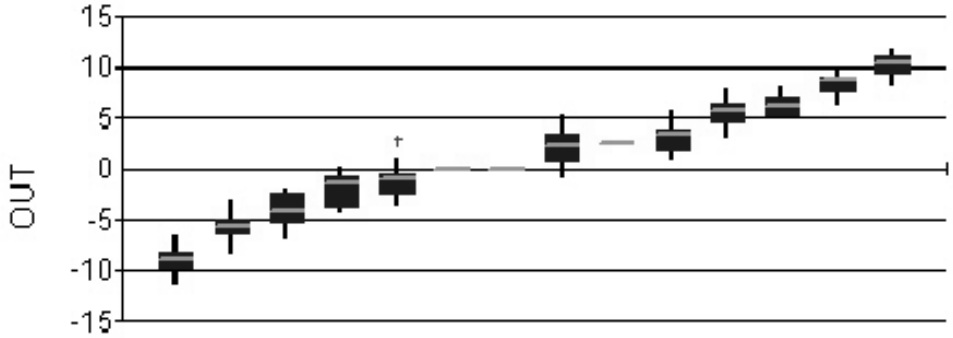


Figure B.12 : Box plots of terminal nodes sorted by target variable prediction

Figure B.12 shows that there will be some inaccuracies, especially between -1 and -5, because the variability is relatively large. A similar remark can be made for values around 2.5.

The predictions done by the 1SE tree in Figure B.11, and by the 5UNN method, are depicted in Figure B.13, which shows a reasonably good fit for both methods.

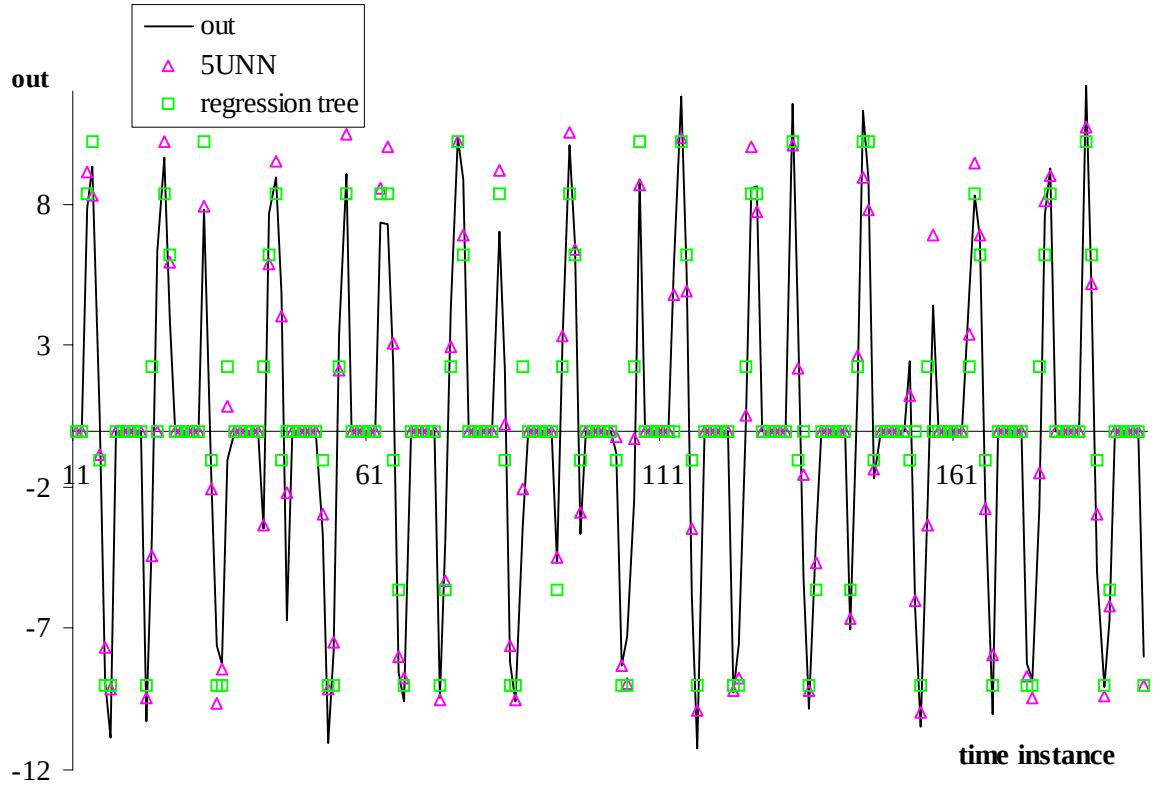


Figure B.13 : Predicted values by the 5UNN method under the optimal mask and by the optimal regression tree

SSE		SAE	
5UNN under optimal mask	Regression tree	5UNN under optimal mask	Regression tree
202.4	655.2	101.7	184.5

Table B.7 : SAE and SSE values for 5UNN and regression tree prediction

Table B.7 shows that the 5UNN method performs better than the regression tree, but it is not an order of magnitudes different. The better performance of the former is to be expected when comparing a lazy and eager method. The differences become more distinctive when considering Figure B.14.

Figure B.14 is in concordance with Figure B.12. It shows the worse prediction for output values between -5 and 0 and in the middle of 0 and 5.

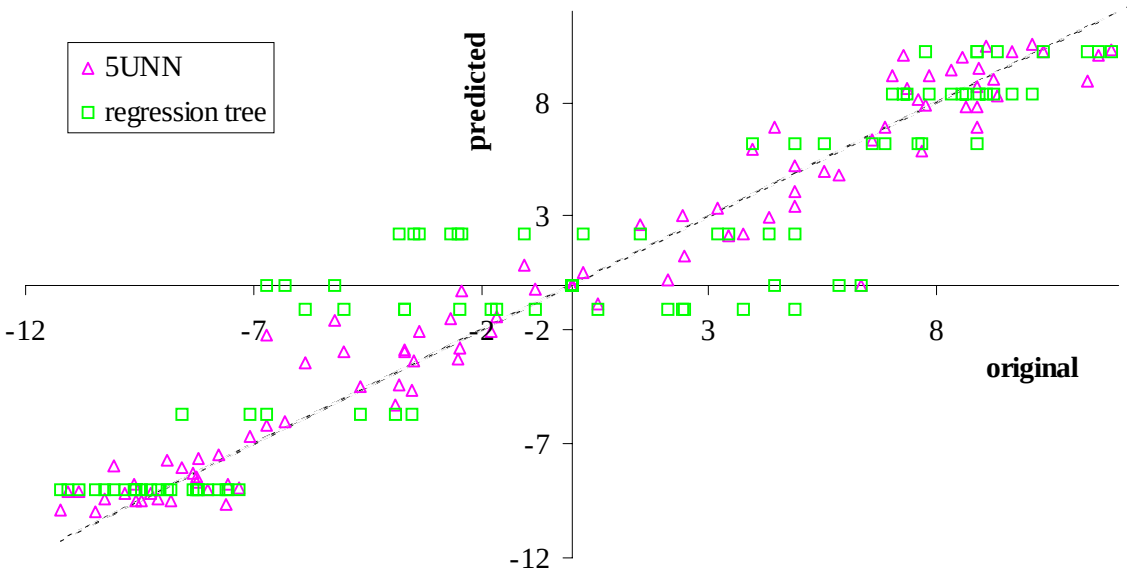


Figure B.14 : Predicted versus original output values for the 5UNN method under the optimal mask and for the 1SE regression tree

B.5 Conclusion

	SSE	SAE
State-observation with sub-optimal mask	3485	455
5+1NN with sub-optimal mask	3723	369
5NN with sub-optimal mask	932	243
5UNN with sub-optimal mask	1011	249
5UNN with optimal mask	202	102
5UNN with second best mask	200	101
Regression tree	655	185

Table B.8 : Overview of the used forecasting methods

Appendix B confirms that the state-observation matrix method of forecasting is not a good performer with regard to accuracy.

In agreement with the findings in appendix A, it can be claimed that the 5UNN method performs at least as well as the 5+1NN.

The prediction performance of the tree regression model is somewhat between the best five nearest neighbour methods with the sub-optimal mask and the 5UNN with the *optimal mask* (which is sometimes not attainable). Hence, regression trees show promising perspectives.

In addition, the example demonstrated that sometimes the second best optimal mask could actually be better than the optimal one, which confirms the subjectivity of the former.

Appendix C

A Real-World Economic System

C.1 Aim of the test

The aim of this experiment is

(a) to see how regression trees perform with regard to forecasting in SAPS,

(b) to investigate what can be done when a trend is present, see section C.3 - C.5,

(c) to find out how high the cardinality of maximal allowable masks may be, see section C.5.2,

(d) to demonstrate how missing values can be tackled with the regression tree approach, see section C.6,

(e) to look if scatter plots may help in determining a pattern a priori, see section C.7,

(f) to illustrate how feedback from the latter approach to SAPS can happen, see section C.8.

For relatively large primary masks, an optimal mask search is out of the question, while a regression tree approach still works. In addition, this example will demonstrate the limitations of a pattern recognition approach.

C.2 Description of data

The data comes from a database of economic data [Crombez 1999]. It stands for the monetary control and the implications for the economy. The records, which describe the change in industrial production, are recorded monthly from February 1959 until February 1999. The first variable is the US changes in consumer prices (CCP), the second variable is US federal funds rate (%) (monthly average FFR), the third variable is US money supply currency (MSC), while the output is US industrial production (IP). From 481 records, 321 were retained for training.

Inputs: CCP, FFR, MSC (continuous)

Output: IP (continuous)

Figure C.1 shows a plot of each variable versus time. Finally, Figure C.2 shows the output. Remark the trend in the MSC and IP variables.

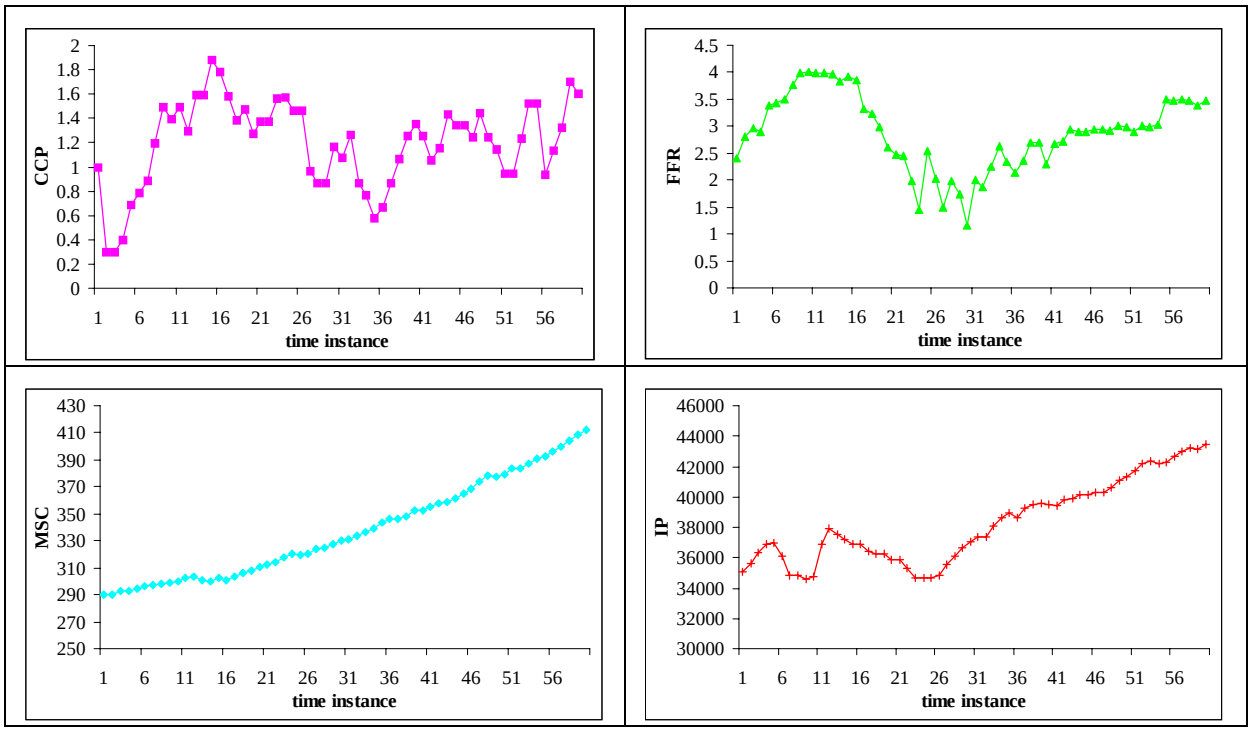


Figure C.1 : First 60 records of the four variables

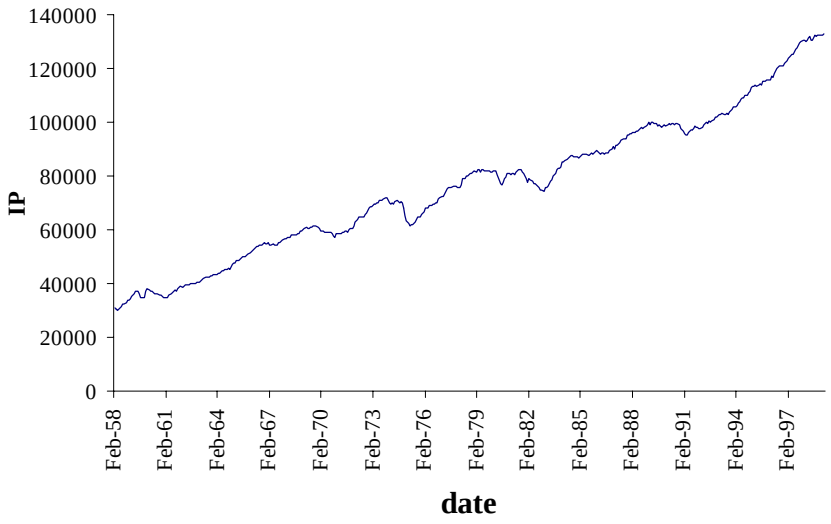


Figure C.2 : IP (output) in function of time

The data that is offered to SAPS-ST consists not only of the three input variables CCP, FFR, and MSC, but also of the derived variables DCCP, DFFR, and DMSC which are the first order differences. This is to see if a relationship includes the 'derivatives' of the original input variables.

C.3 Looking for a pattern in the original data (without detrending)

A pattern recognition approach is based on the recognition of existing patterns in the training set. These approaches cannot extrapolate outside what is observed at least once. Hence, when a trend is present it becomes difficult to use the pattern recognition approach to find a predictive model. The next subsection illustrates this limitation, even under more idealised circumstances where first order detrended input variables are added (data-augmentation)

C.3.1 Sub-optimal mask searching

Gaussian uniform three-levelled recoding for each variable is done. A maximal allowable mask with a memory depth of 12 (1 year) is used. In this experiment, CCP, FFR, MSC, DCCP, DFFR, and DMSC are inputs for determining IP. It took the prototype quite some time (approximately 3 days) to find the sub-optimal mask. It has a quality of 0.4876 and it corresponds to

$$IP(i) = \tilde{f}\,(MSC(i-12), DFFR(i-11))\,.$$

Figure C.3 shows the very bad forecasting with the 5+1NN method on the test set.

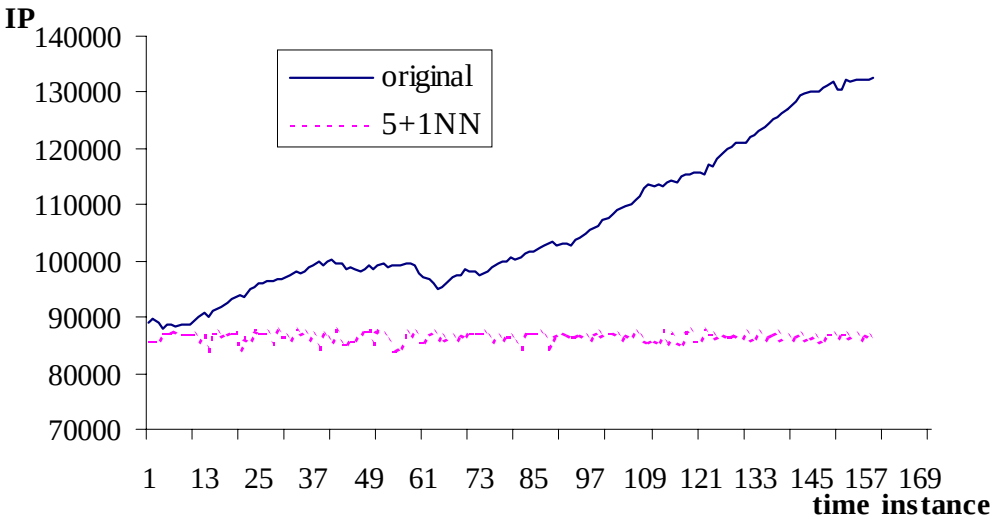


Figure C.3 : Predicted (via 5+1NN) and original values for the output IP

C.3.2 Using regression trees on non-detrended data

The former experiment showed that, under a trend, the pattern-recognition approach fails. Regression trees are from another paradigm: they partition the measurement space. This experiment will illustrate that regression trees can model a trend. Here, it is opt to take the original (non-detrended) inputs and output, but to apply a maximal allowable mask of memory depth 24. Hence, the inputs are now CCP, FFR, and MSC. A simple tree with 13 terminal nodes is selected, because its cross-validation error is just below 1% (it is 0.9 %). A prediction on the training set is shown in Figure C.4. Hence, the regression tree can model the *training set* very well with a relatively simple model.

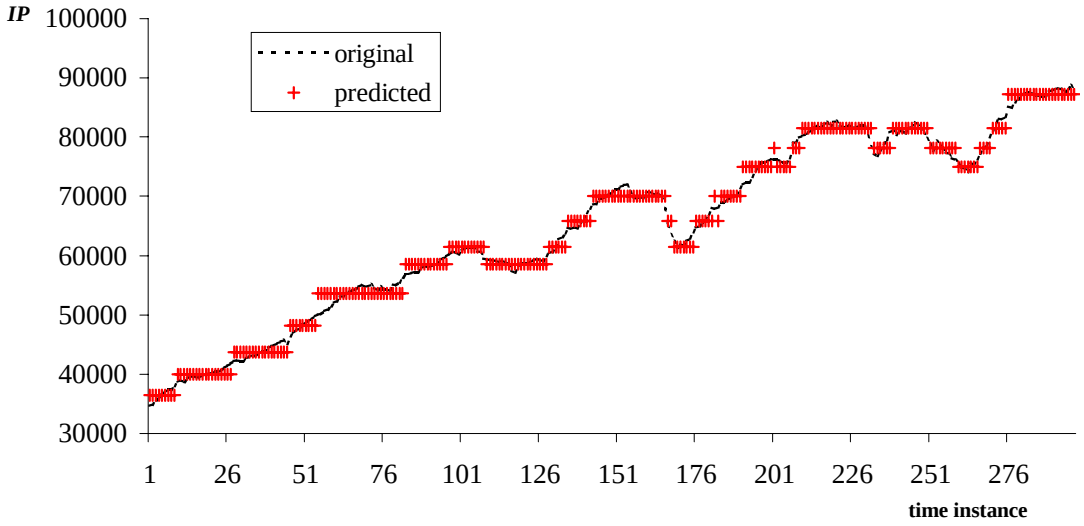


Figure C.4 : Predicting on the training set with a depth of 24 months

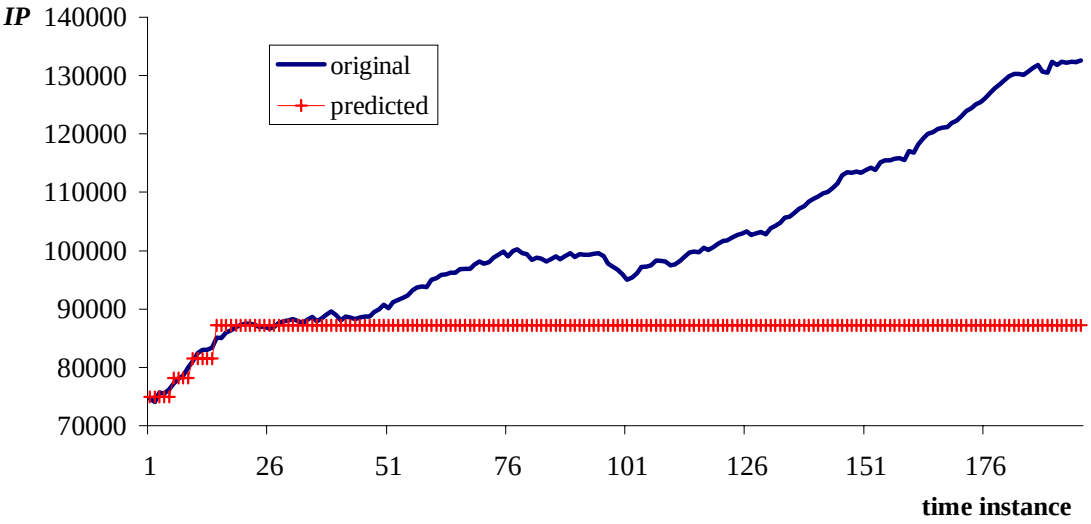


Figure C.5 : Predicting on the test set with a depth of 24 months

However, *test set* predictions, displayed in Figure C.5, clearly show that the fit is not good at all, except for some initial points, which are in common (overlap) with the training set.

C.3.3 Conclusion

To be able to find a pattern, it has to occur in the training set (be it disguised or not). Neither the sub-optimal nor the data-mining approach can cope with a trend. A possible and general solution to improve forecasting is to take differences of the output to remove the trend from the output. The same is done for the inputs, and just to play it safe, they are all kept (original and detrended inputs) in the maximal allowable mask. Section C.4 elaborates on this approach.

A less general solution is to remove the trend by trying to fit all kind of functions. In that case, one usually tries polynomial functions. Section C.5 does exactly that.

C.4 Detrending by first order differencing

In this section, a good model is searched after first order detrending of the output variable *IP*, i.e., by creating the detrended variable *DIP*, given by $DIP = IP(i) - IP(i-1)$). Using the same recoding and maximal allowable mask as in section C.3.1, SAPS-ST finds the sub-optimal mask with a quality of 0.2088, which corresponds with

$$DIP(i) = \tilde{f}(CCP(i-11), DCCP(i-10), DCCP(i-12), DIP(i-11))$$

The low quality is due to the low degree of determinism (0.313) and relatively low observation ratio (OR = 0.667). The lowest SSE and SAE are found for the 5UNN method. Therefore, only this one is put in Figure C.6, which shows the original and predicted values.

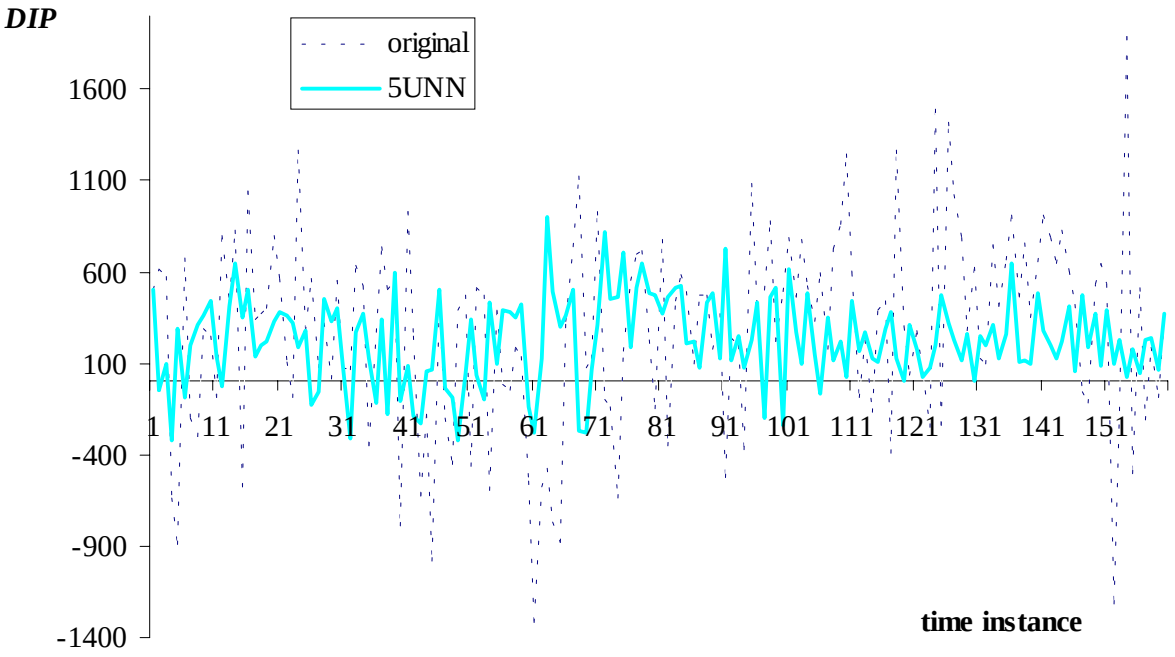


Figure C.6 : Forecasting and original values for the output DIP

One sees that the prediction appears to be bad. Hence, using first order differences does not help.

Note: Another attempt with a recoding of five levels for each variable was tried. The resulting optimal mask (with a quality of 0.2509) did not improve the forecasting significantly.

A last attempt was tried with a regression tree on the same maximal allowable mask, but to no avail. When a regression tree is grown on the training, the cross-validated relative error starts increasing again too soon for a regression tree curve (see Figure C.7). This indicated an overall poor prediction.

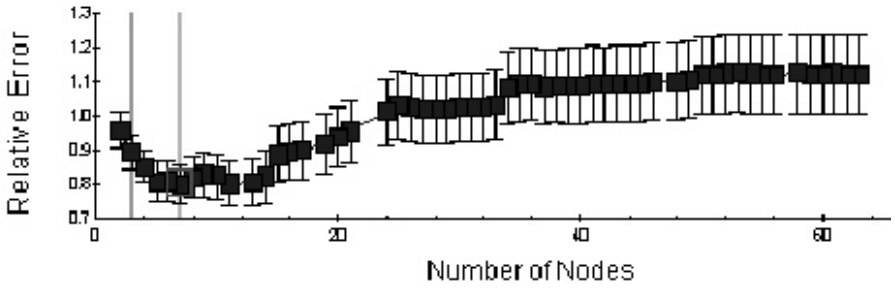


Figure C.7 : Growing a tree on DIP for a maximal mask depth of 12

Using the minimum cost tree on the test set gives a very bad prediction. This is shown in Figure C.8.

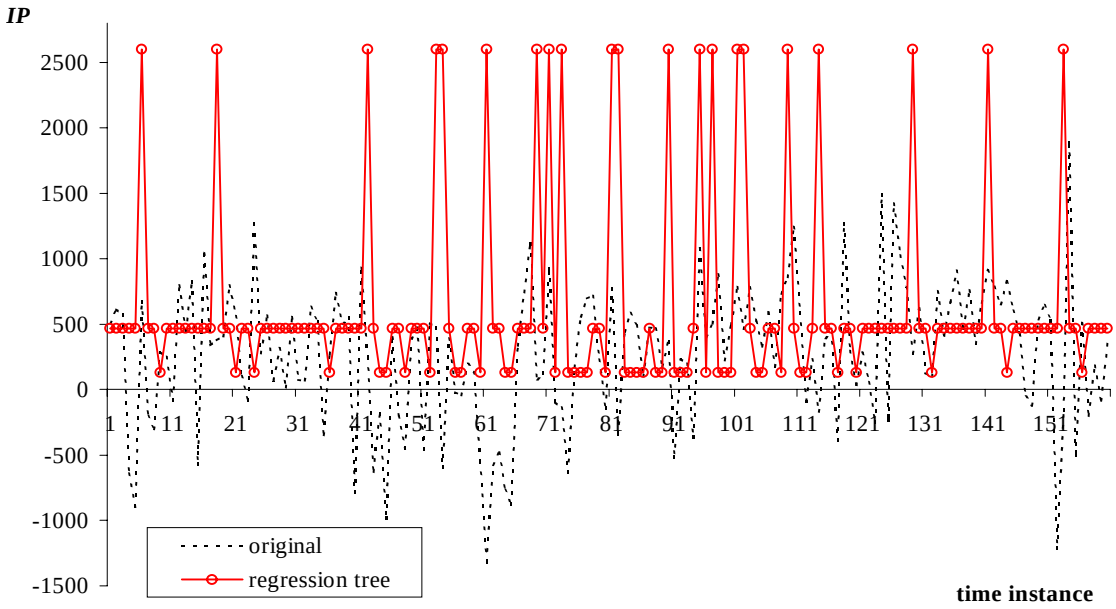


Figure C.8 : Original and prediction values of DIP

Conclusion about the differencing

The sub-optimal approach does not find a good prediction model for memory depth up to 12 time steps (months of 1 year). Two reasons can be found for this:

- there is no pattern for the given data set
- the maximal allowable mask is not deep enough

Some deeper maximal allowable masks were tried (memory depth of 36) with the regression tree approach, but to no avail. This suggests that there is no recognisable pattern in the given data set up to memory depths of 36 deep.

There is still another option that can be tried to remove a trend. It consists of fitting a linear or higher order curve, and to subtract the fit curve from the original signal. The residual signal is then kept for subsequent pattern recognition. This approach is described in section C.5

C.5 Detrending with a polynomial fit

The original data were read in MATLAB© (see [MATLAB 1999]) and a linear and/or quadratic fit has been used to remove the trend. All variables were investigated with MATLAB to see if there was a trend[1], but only *MSC* and *IP* were found to contain one. Hence, it was only necessary to detrend the variables *MSC* and *IP*. Their detrended curves are displayed in Figure C.9. *MSC* is detrended by a quadratic curve and *IP* by a linear curve. Hence, the detrended variables are denoted respectively by msc2d and ip1d. The detrending is done on the whole data range, while it should normally be done only on the training set. Luckily, this will not change the end result much (only a very small positive bias for prediction errors may be introduced).

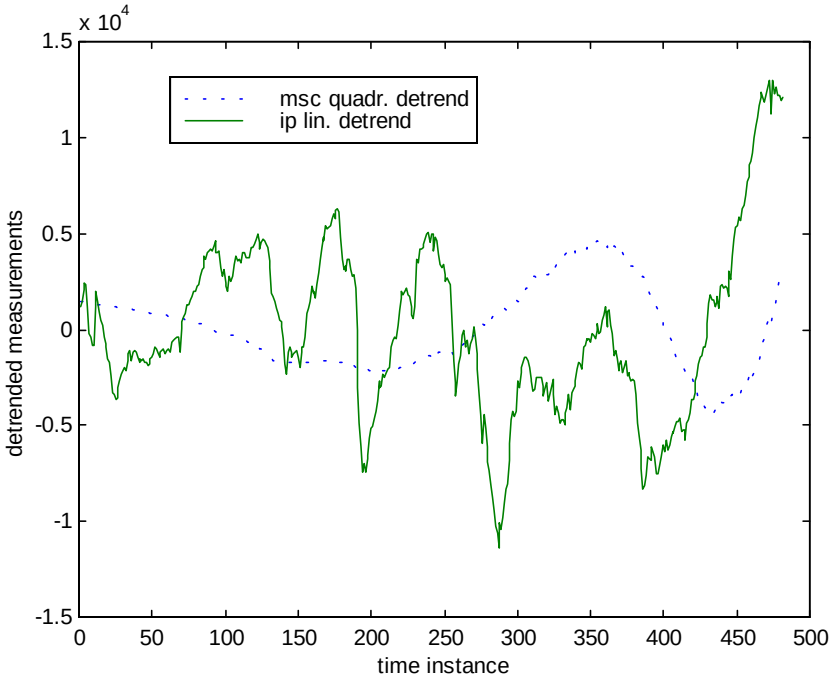


Figure C.9 : Detrended curves

The detrended data were then used to find a pattern. However, it is doomed to fail (at least partially). This can been seen with the aid of Figure C.9 that already shows why the prediction will go wrong for the latest time instances (although all data is used to bias the linear and quadratic trend removal). The curve for the detrended output variable ip1d shows in its last part something that is not present in the first 2/3 of the curve. Therefore, one cannot expect the pattern-recognised paradigm to find any pattern for predicting this phenomenon in the very last part of the test set.

C.5.1 Comparing the effect of the detrending method

One might expect that the previous 'strange' relative error curve behaviour is due to the limited mask depth. Figure C.10 shows that this is not true. The new detrended variables ip1d1 and msc2d are used, but the maximal allowable mask is the same as in Figure C.7. Now, the curve has a normal decay in the relative error curve. Hence, the detrending method does matter. More research is needed to dig for its underlying cause(s).

[1] This has been achieved by using the MATLAB functions *polyfit* and *detrend*.

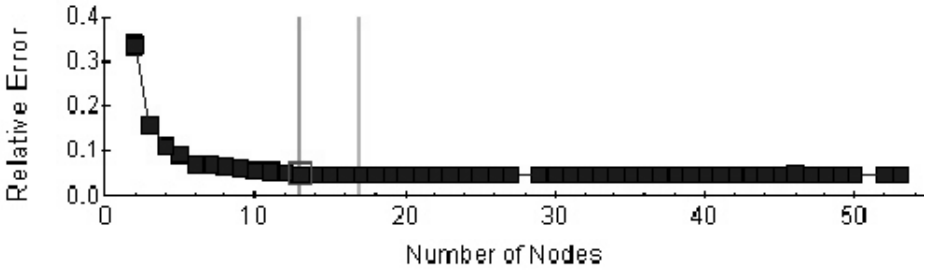


Figure C.10 : Growing a tree on ip1d for a maximal mask depth of 12

C.5.2 The pessimistic approach: using a very deep maximal allowable mask

The first thing one could investigate further is if a prediction model can be found if the maximal allowable memory depth is increased to, say 24 months, or even more. However, in the latter case, one sees from the small simulation that the prototype is too slow to perform very deep mask evaluations. Even when rewriting the prototype, such that the average computing time per mask decreases, it will still have its limitations because evaluation of a mask involves a lot of matrix manipulations (to generate the state-observation matrix). So even the sub-optimal mask search has its limitations. Consequently, regression tree approach will be used to look for masks with memory depth larger than 12.

For a maximal allowable mask of 60 deep, a 1SE tree with 13 terminal nodes (and 5% relative cross-validation error) is found. The variability in the terminal nodes is depicted in Figure C.11.

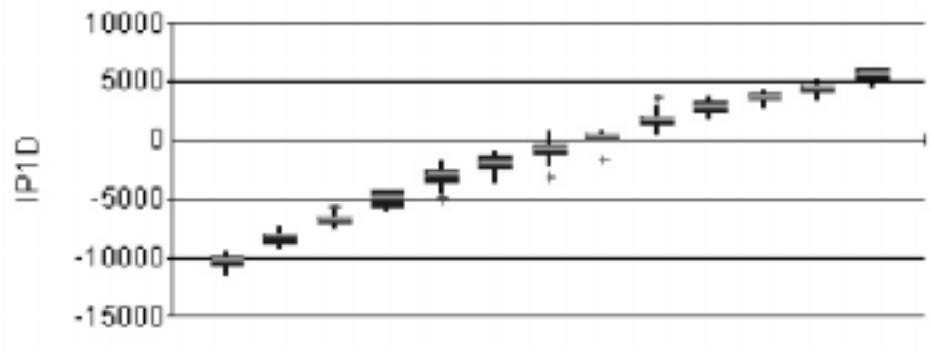


Figure C.11 : Terminal nodes sorted by target variable prediction

The importance of variables is given by Table C.1 and Table C.2. The former incorporates contributions of variables as first surrogate. The latter only considers primary splitters.

Ranking	1 (very high)	2 (very high)	3 (very low < 3%)
	ip1d(*i*-1)	ip1d(*i*-2)	CCP(*i*-31), ip1d(*i*-48), …

Table C.1 : Variable importance (includes role as first surrogate)

Ranking	1(very high)	2 (very low, < 3 %)	3 (extremely low, < 1 %)
	ip1d(*i*-1)	ip1d(*i*-48)	ip1d(*i*-50), ip1d(*i*-18), msc2d(*i*-36)

Table C.2 : Variable importance when only primary splitters are considered

The information from Table C.2 is visible in Figure C.12. The latter shows in addition where the primary splitters are used in the tree.

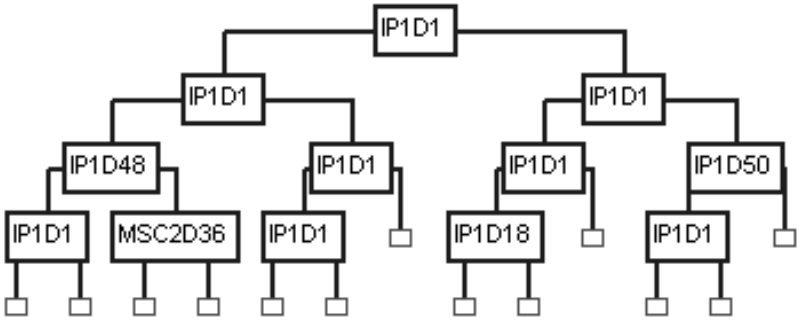


Figure C.12 : The regression tree and its splitters

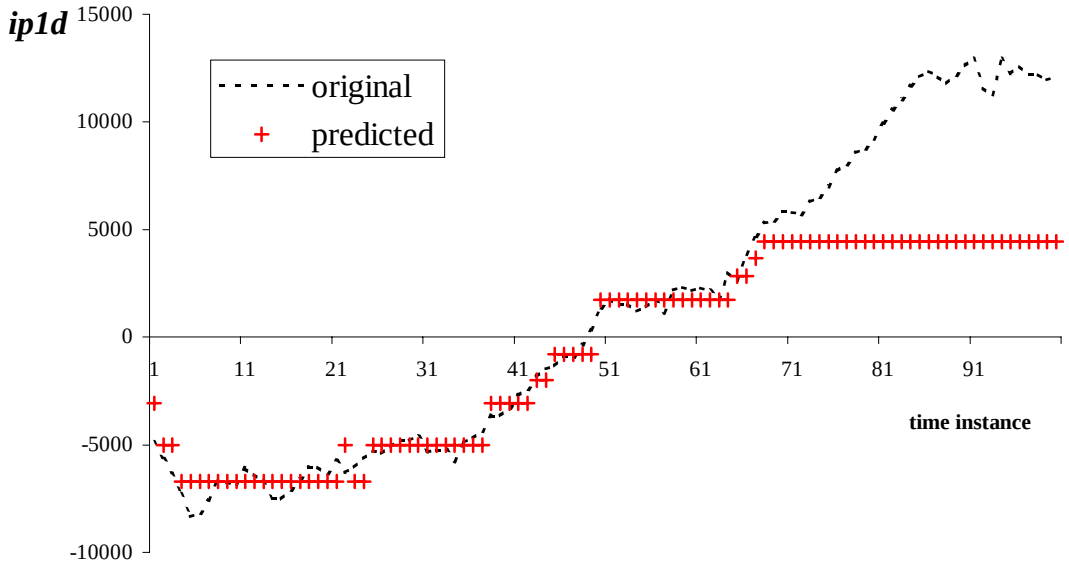


Figure C.13 : Original and predicted output ip1d on test set

Figure C.13 illustrates the bad prediction for the last years.

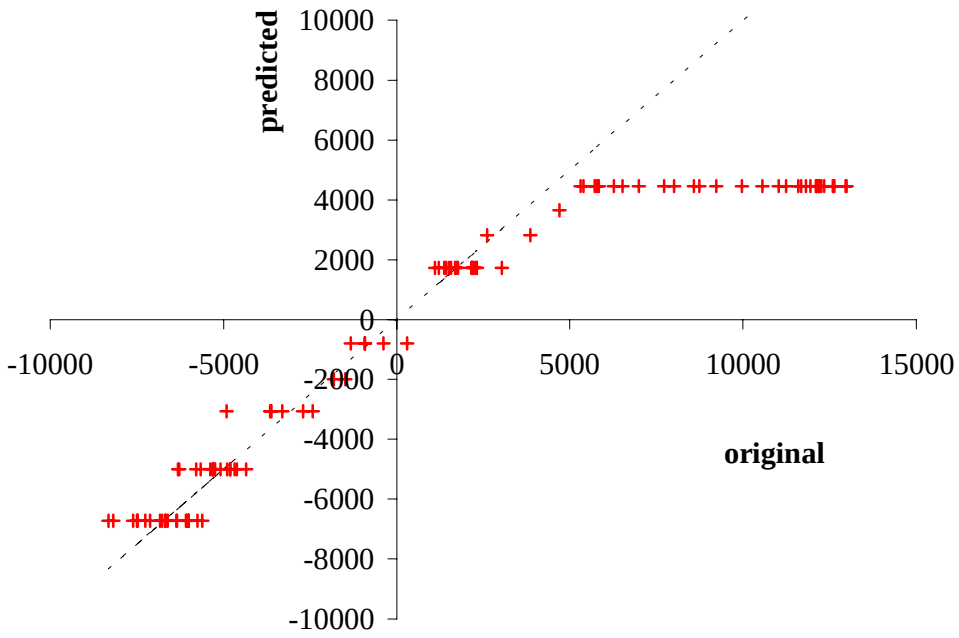


Figure C.14 : Predicted versus original output ip1d

The bad forecasting performance for large values, which appear at the end of the test set, is also visible in Figure C.14. This is to be expected from Figure C.9.

C.6 Using missing values

When some values are missing, the regression tree still performs very well. As an experiment, some values from the variable ip1d(*i*-1) were removed directly from the static matrix. This variable was the first one on which splitting occurs. Consequently, the worst possible case is simulated. Figure C.15 shows that a lot of missing values are present in the test set.

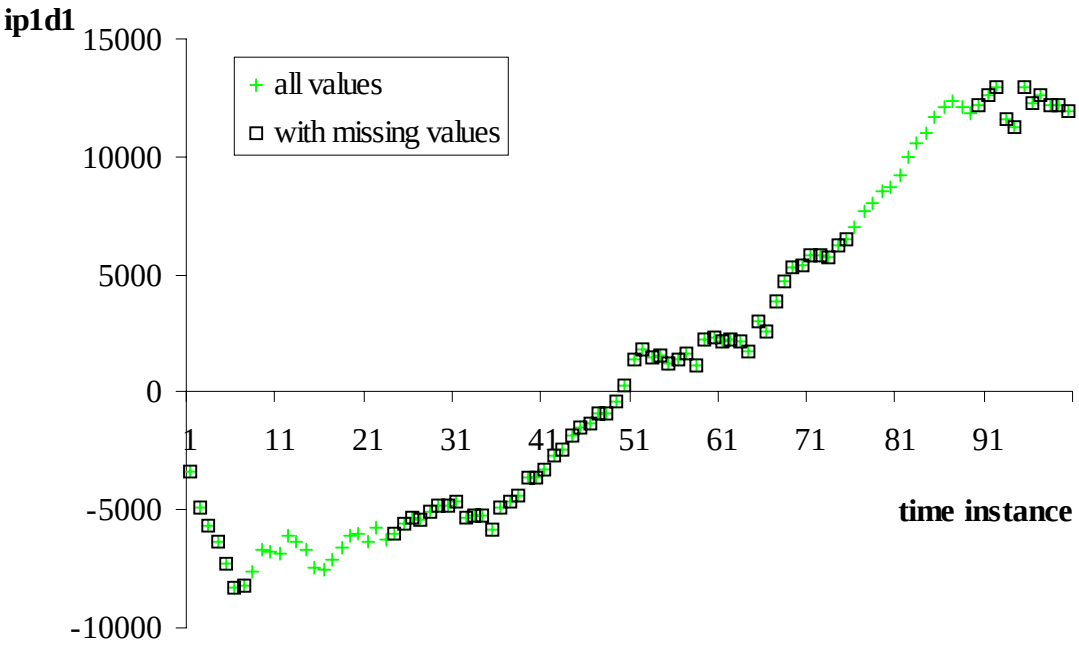


Figure C.15 : 'Inserting' missing values

Figure C.16 displays a reasonably good forecasting.

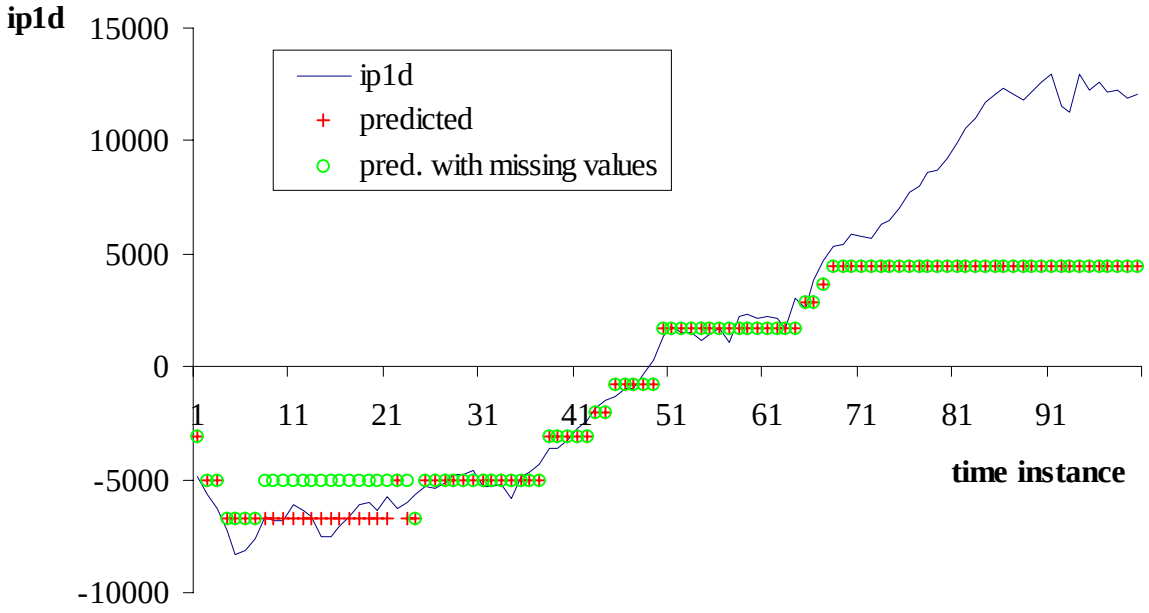


Figure C.16 : Prediction with missing values

All data		Missing values	
SAE	SSE	SAE	SSE
216995	$1259\ 10^6$	233534	$1295\ 10^6$

Table C.3 : SAE and SSE for prediction with all data and with missing values

Table C.3 confirms the finding that the prediction with the missing values do not suffer much from the missing values. Hence, the surrogate split performs very well.

C.7 Trying scatter plots to have a hint about patterns

The sought patterns can be hidden in the dependencies between the output and multiple inputs. Scatter plots, as in Figure C.17 until Figure C.20, can be tried to detect simple patterns. However, no clear pattern can be seen (maybe FFR, MSC and/or DMSC with regard to IP?).

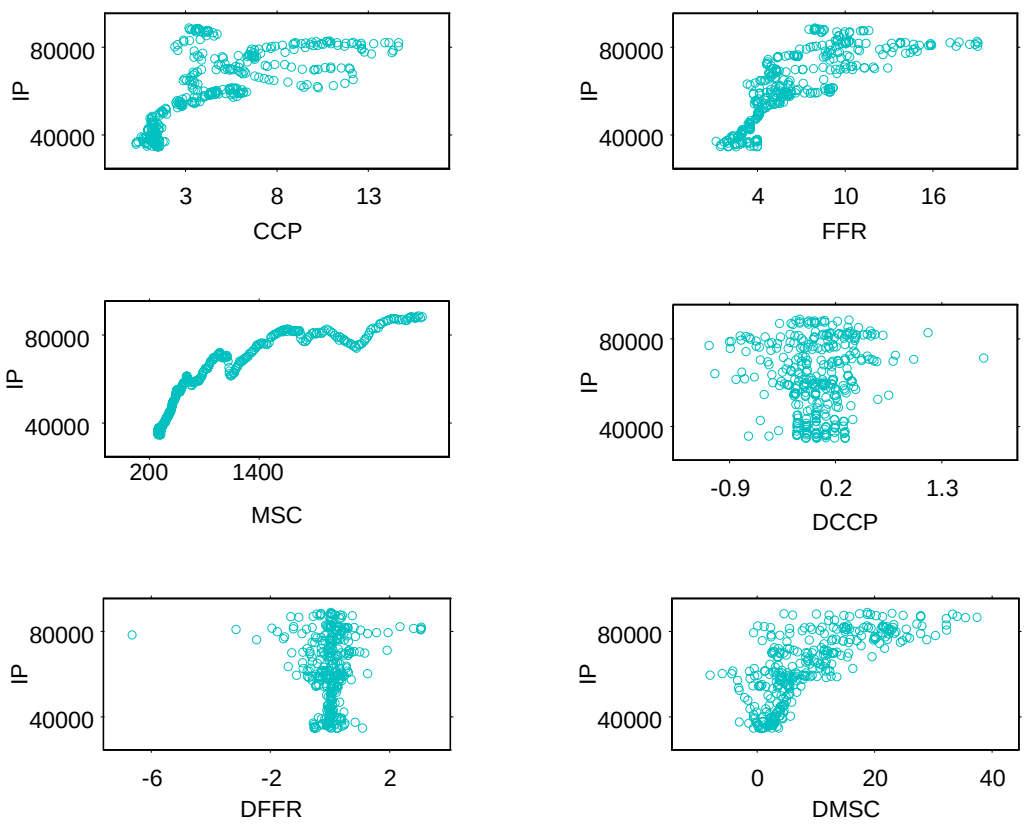


Figure C.17 : Scatter plots of IP versus the inputs (no detrending for output)

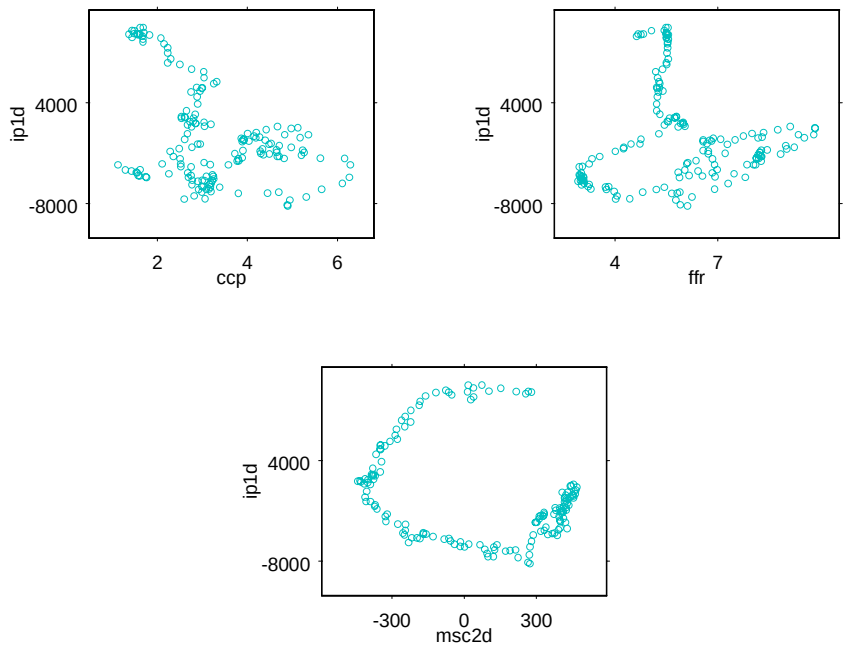


Figure C.18 : Scatter plots of output ip1d (linear/quadratic detrended data)

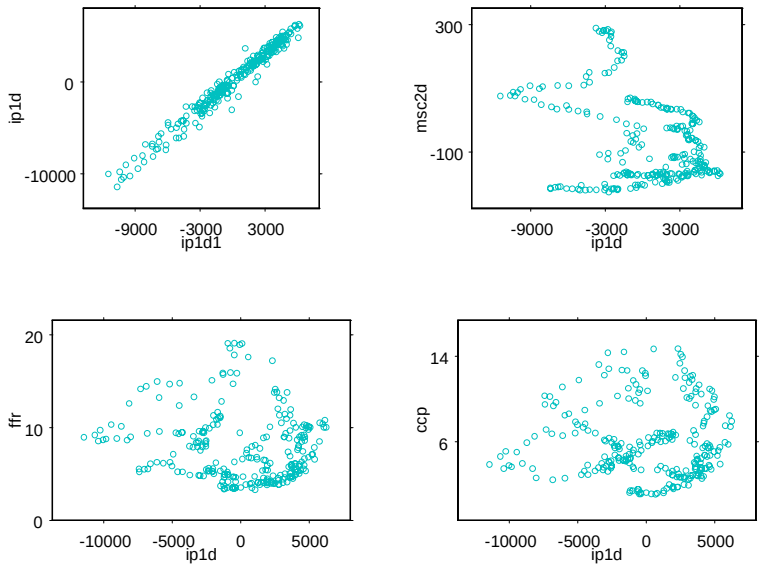


Figure C.19 : Looking for patterns in the state space (part 1) for ip1d

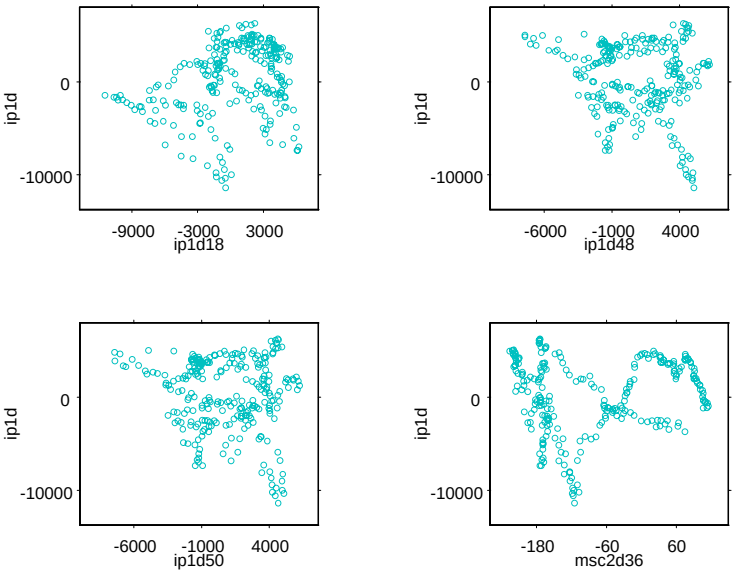


Figure C.20 : Looking for patterns in the state space (part 2) for ip1d

Conclusion

If a splitter is near the top of the tree, then one sees a pattern that is easily recognizable by regression tree in the scatter plot. Deeper in the regression tree, the pattern cannot be seen from the scatter plots anymore, because of their context and locality.

C.8 Feedback from the data mining method to SAPS

The information contained in Figure C.12 (or Table C.2) with regard to the splitters used in the tree induction, can be used to compose a mask in SAPS. The corresponding mask is given in Table C.4. The ranking information gets lost when performing the feedback into a primary mask.

Reference index	CCP	FFR	msc2d	ip1d
-50	0	0	0	-1
-49	0	0	0	0
-48	0	0	0	-1
…				
-36	0	0	-1	0
…				
-18	0	0	0	-1
…				
-3	0	0	0	0
-2	0	0	0	0
-1	0	0	0	-1
0	0	0	0	1

Table C.4 : Generating a mask via data mining

Hence, the mask in Table C.4 can be used in SAPS as a primary mask on which an optimal search is done (the recoding of the data is ccp (5G), ffr (5G), msc2d (3G) , ip1d (5G)). The resulting optimal mask has a quality of 0.696 and corresponds with the very simple relationship given by

$$\text{ip1d}(i) = \tilde{f}(\text{ip1d}(i-1))$$

The second best optimal mask has a quality of 0.394 and corresponds with

$$\text{ip1d}(i) = \tilde{f}(\text{ip1d}(i-36), \text{ip1d}(i-1))$$

The prediction with the optimal mask for the 5+1NN and the 5UNN method is shown in Figure C.21. Their forecasting performance is approximately the same. It shows the same good forecasting as with regression trees.

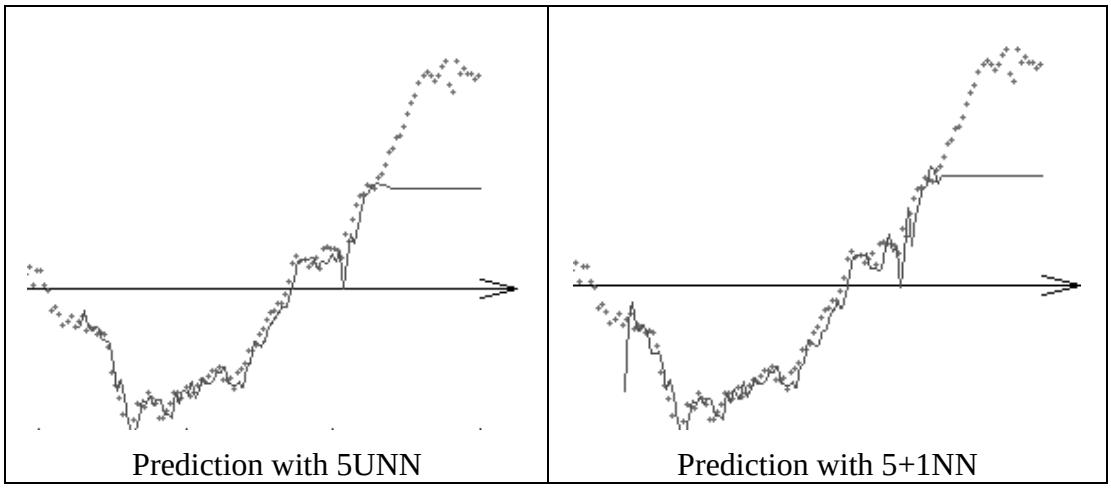


Figure C.21 : Forecasting on the test set

This is also expressed in Table C.5 and Figure C.22.

SSE		SAE	
1+5NN	5UNN	1+5NN	5UNN
$764\ 10^6$	$822\ 10^6$	$176\ 10^3$	$175\ 10^3$

Table C.5 : SAE and SSE for 5NN after feedback from CART

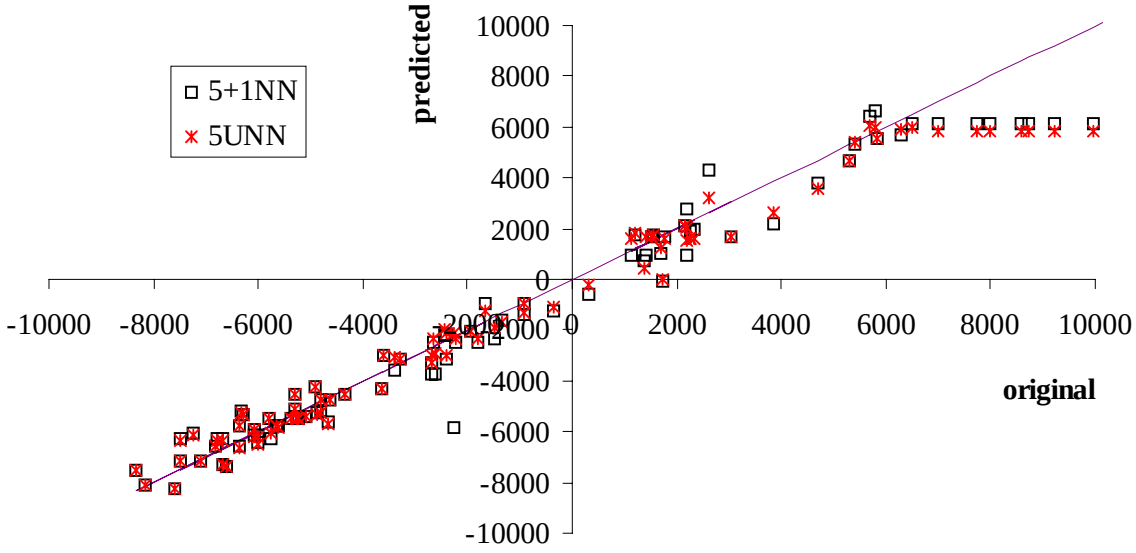


Figure C.22 : Predicted versus original values after feedback from CART

C.9 Conclusion

When detrending with a linear trend for the output IP, and a quadratic for the variable MSC, a good and very simple predictive model can be obtained. The new regression tree approach allows quite large maximal allowable masks (here 60 deep) to start with, which were simply impossible with the SAPS-II version. The approach is quite fast in execution time and it finds good models when appropriate pre-processing is done. However, it cannot predict unseen patterns, but this is because of the used pattern recognition paradigm.

Missing values can be treated very well in regression trees. A regression tree can also be used to give feedback to SAPS. This appendix demonstrates this possibility. In this case, the resulting optimal mask was very simple, but this may not necessarily always be the case.

Appendix D

A Real-World Water Demand System

D.1 Aim of the experiment

The water demand data set comes from the team of Cellier [1999]. It consists of data gathered from a water distribution system in Portugal. There are six points where water is taken (at Cotao, Mabrao, Merces, Pimenta, Ranholas, and Rinchoa). This example demonstrates that very entry-complex masks are possible for data sets with quite some records. Unfortunately, the limitation of the CART version used in the regression tree approach will limit the number of entries in a mask. Therefore, a way to copy with these limitations is illustrated. It also shows how one can go beyond the initial maximal allowable mask. Hence, the aim of this experiment is

(a) to illustrate the regression tree approach for a large databases of records.

(b) to show how deeper masks can be tackled,

(c) to demonstrate that deeper masks not always give a better forecasting,

It is not the aim of this appendix to exploit the data completely by trying all kind of masks.

D.2 Setup and data generation

There are 16 variables, each with 13128 records. The data is read in from different files with MATLAB and merged into a single matrix. The inputs consist of

- The water demand in Mabrao is stored in the variable d1.
- The water demand in Pimenta is stored in the variable d2.
- The water demand in Cotao is stored in the variable d3.
- The water demand in Ranholas is stored in the variable d4.
- The water demand in Rinchoa is stored in the variable d5.
- The water demand in Merces is stored in the variable d6.
- The pump data is stored in two variables (two pumps), n2 and n3 respectively (no data about pump1 is available).
- The data about the valves is stored in u1, u2, u3, u4, u5, u6 and u7.

The output is the pressure *p* at a certain point in the distribution network.

Hence, there are 15 inputs and 1 output. The system works within a certain operating range, so no trends are expected a priori (in contrast to appendix C). The whole set of observations for the output is depicted in Figure D.1. The first 48 records for all signals are shown in Figure D.2.

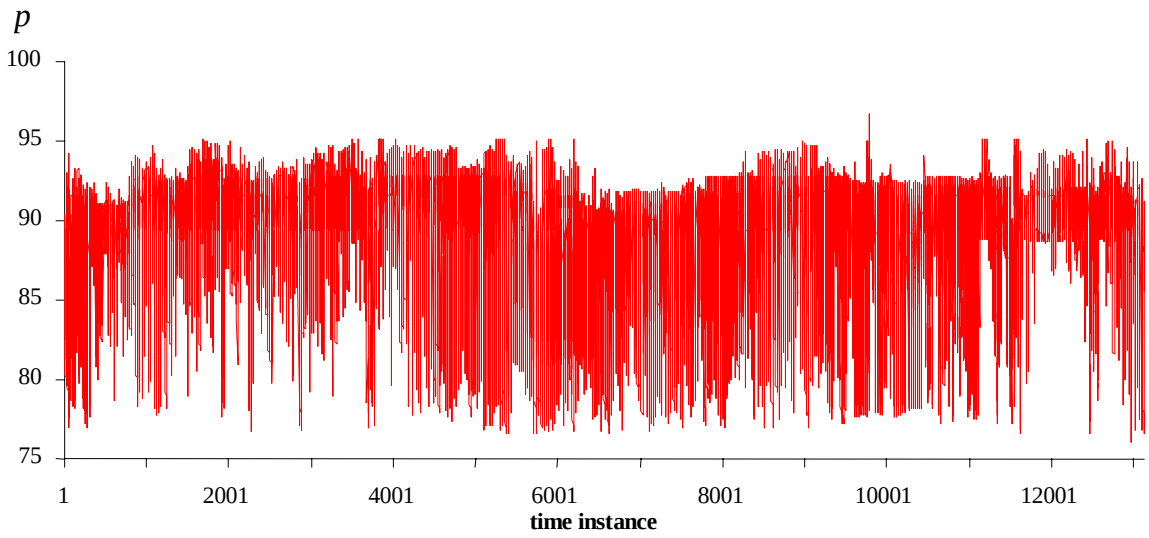


Figure D.1 : Variable *p*

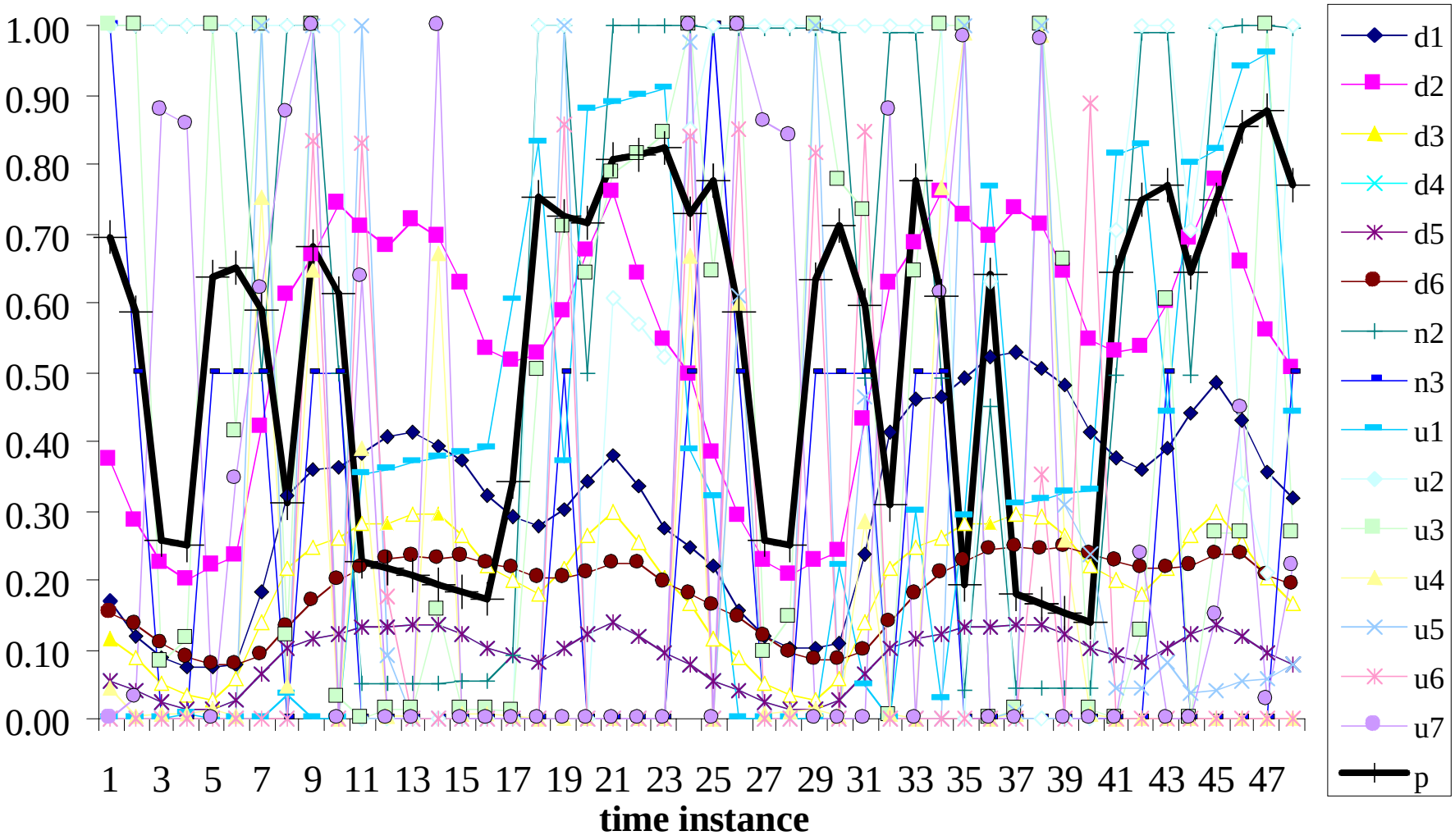


Figure D.2 : Zooming in on the first 48 records (each variable is rescaled to [0,1])

Data mining approach with maximal allowable mask of memory depth seven

As the records are very long and there are many variables, the limitations of the CART version begins to count. The maximum memory depth of a maximal allowable mask is determined by the dimensions of the raw data matrix, because the CART version used here (version 3.6 for 32MB) uses no more than 32MB RAM. The CART documentation says that this corresponds with approximately handling one million entries. Hence, knowing that the cardinality *c* of a maximal allowable mask with a depth *d* for *n* variables is given by

$$c = n \times d$$

The number of mask entries is also the number of state variables in the static matrix that is handed over to CART for subsequent processing.

The training set consists of 2/3 of the total set of 13128 observations. Hence, the total number of entries in the data matrix given to CART is

$$13128 \times \frac{2}{3} \times c$$

Filling in *n* by 16 and setting the total number of entries equal to the maximum number of entries, which is approximately 1000000, *d* is obtained from

$$13128 \times \frac{2}{3} \times 16 \times d = 1000000$$

Solving for *d* gives a maximum depth of 7.14. Therefore, a maximal allowable mask of memory depth 7 is tried, because it may be still feasible for the software package. When applying a maximal allowable mask with a memory depth of seven, one obtains an optimal tree (within one standard error) consisting of 114 terminal nodes (and a relative error of 1.1%). No cross-validation is used (the number of records is above the standard threshold value of 3000 for CART), so a (randomised) fraction of the data is used for testing the internal model validity. This fraction is 0.3333, so 2/3 is used for the learning set and 1/3 for the (internal) test set. The train-test paradigm for backward pruning the tree applies. Consequently, two box plots can be drawn: one for the training set (in the training set), which is depicted by Figure D.3, and one for the test set (in the training set), which is depicted by Figure D.4. The latter replaces the role of the cross-validation error determination.

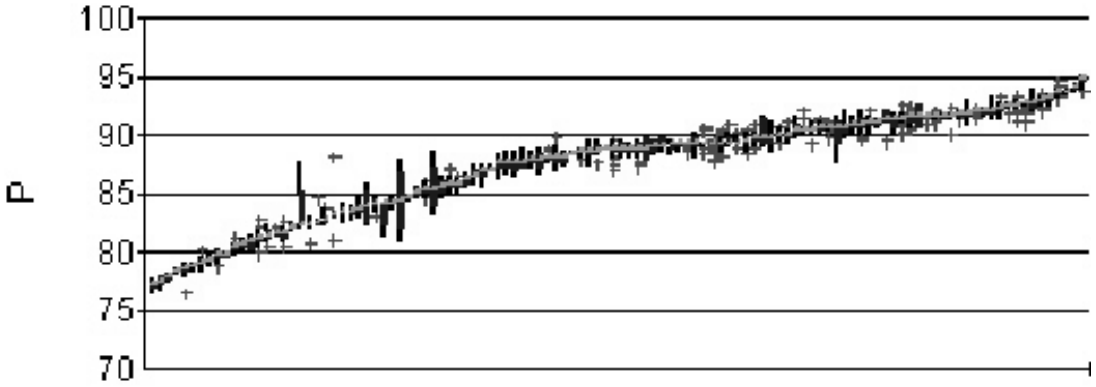


Figure D.3 : Terminal Node Box plots for the learning set

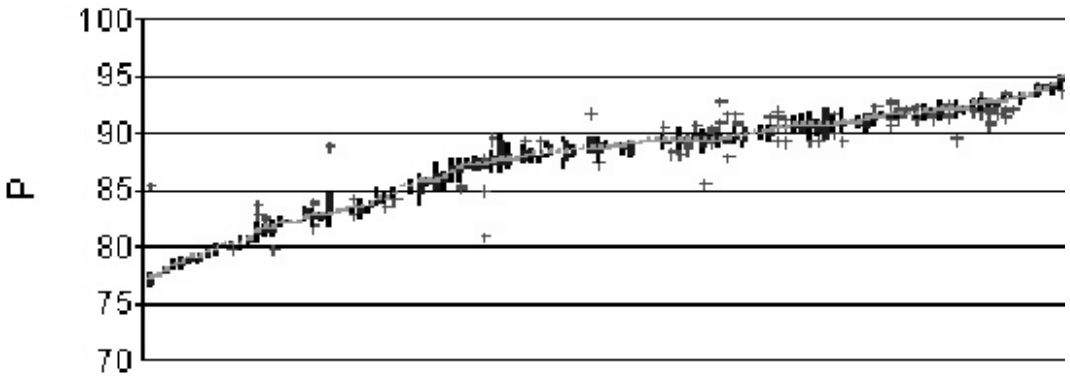


Figure D.4 : Terminal Node Box plots for the (internal) test set

The variable importance, with the top 1 surrogate splitters included, is depicted in Table D.1.

Ranking	1 (very high)	2 (high)	3 (low)	4 (very low < 10 %)	5 (extremely low)
	n2	u2	u1	u1(*i*-2), u4	u1(*i*-3), u1(*i*-1), …

Table D.1 : Variable importance ranking (with first surrogates)

The variable importance, with only the primary splitters included, is depicted in Table D.2

Ranking	1 (very high)	2 (low)	3 (low)	4 (extremely low)
	n2	u1, u1(*i*-2)	u4	n3, …

Table D.2 : Variable importance ranking for primary splitters

Table D.1 and Table D.1 show that n2 is very important, while u2 and u1 play an important role as surrogate. The forecasting on the originally split-off test set is depicted in Figure D.5, which zooms in on the first 400 records.

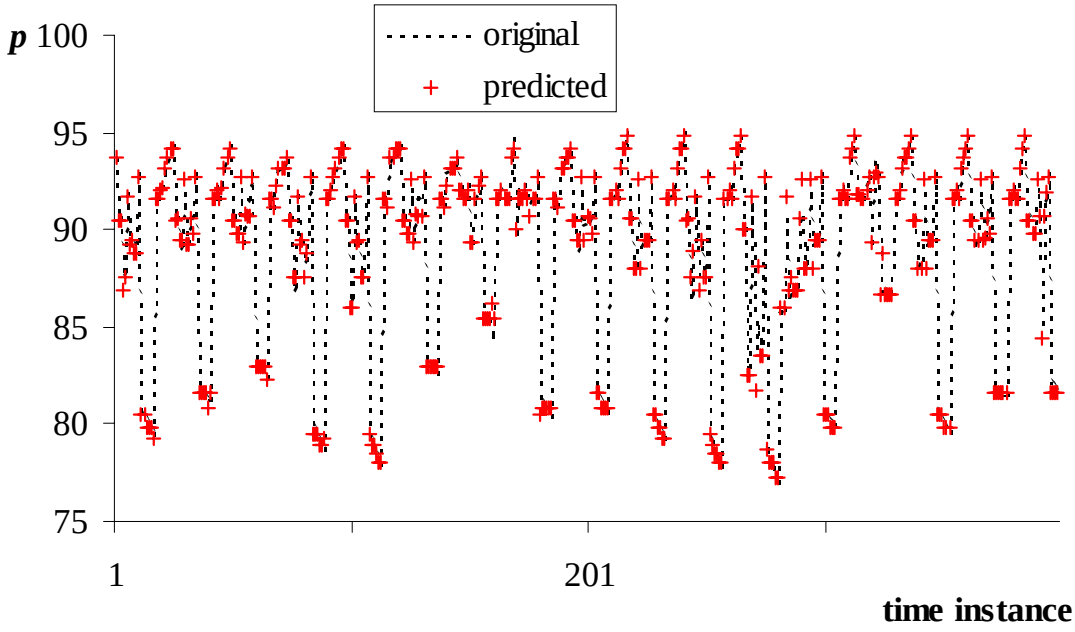


Figure D.5 : Predictions under the primary mask with memory depth seven (first 400 observations)

Comparing the predicted values with the known values in the originally split-off test set is done by the scatter plot in Figure D.6. It shows that the fit is reasonably good.

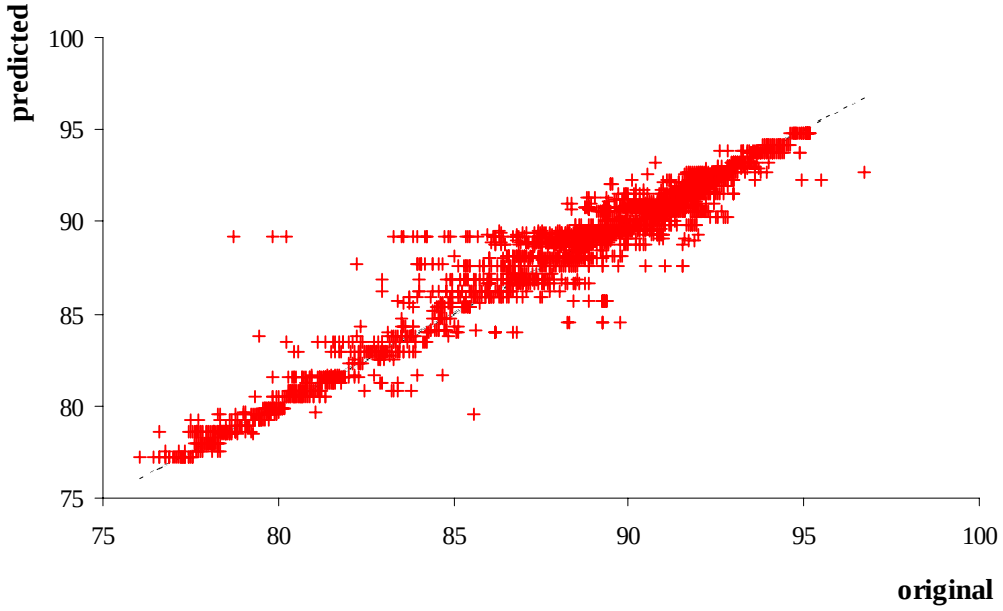


Figure D.6 : Predicted versus original values for the regression tree based on a maximal allowable mask of memory depth 7 (MSE = 0.742 and MAE = 0.464)

D.3 Introducing a gap in the maximal allowable mask

The most logical thing to do, would be to make the mask as deep as possible. Unfortunately, this cannot be done because of the limits of the bought CART version. An alternative is to use a primary mask that has a gap in time. An example is the primary mask listed in Table D.3.

The motivation for this is based on a common sense idea that says that recurring patterns may happen over 24 hours. Therefore, the mask is chosen as such to comprise the instance (*i*-24). This is only one possible primary mask with a gap. It serves as an illustration on how one could investigate for deeper patterns. Other primary masks can be devised as well, but they will not be tried in this dissertation.

ref	d1	d2	d3	d4	d5	d6	n2	n3	u1	u2	u3	u4	u5	u6	u7	p
-26	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-24	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1

Table D.3 : Primary mask for second experiment

The primary mask in Table D.3 is chosen such that it includes the states that were found important under the first primary mask (see Table D.1 and Table D.2).

The resulting 1SE regression tree has 122 terminal nodes and a relative error of 1.7%. The variable importance, with the top 1 surrogate splitters included, is depicted in Table D.4.

Ranking	1 (very high)	2 (high)	3 (low)	4 (very low)
	$p(i\text{-}24)$	n2(i-24)	n2, u2, u1	u4, n3, …

Table D.4 : Variable importance ranking (with first surrogates)

The variable importance, with only the primary splitters included, is depicted in Table D.5.

Ranking	1 (very high)	2 (low)	3 (very low)
	$p(i\text{-}24)$	n2	u1, u4, n3, u2, …

Table D.5 : Variable importance ranking for primary splitters

It appears that the output strongly depends on its value, 24 time instants before (if that information is made available through a deeper mask). Comparing Table D.4 and Table D.5 reveals that the variable u2, which was so important in the previous section, plays the role of surrogate.

The forecasting on the originally split-off test set is depicted in Figure D.7, which zooms in on the first 400 records.

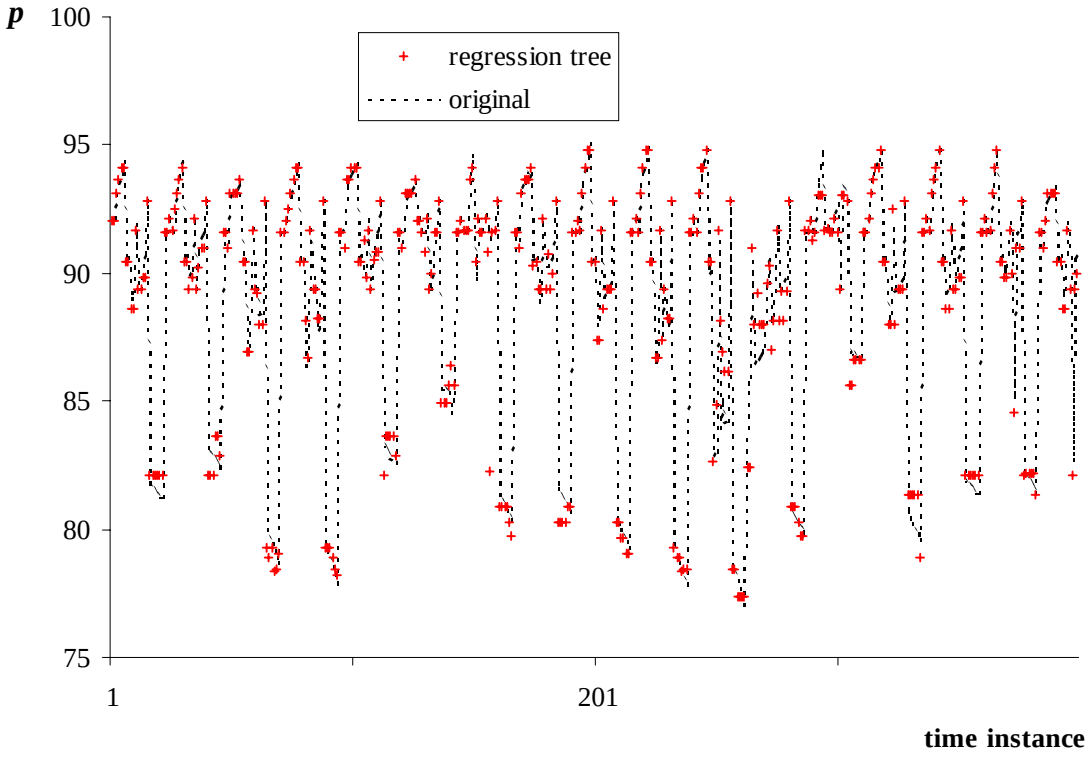


Figure D.7 : Predictions under the primary mask with a gap (first 400 observations)

The predicted versus original values are shown in Figure D.8. It shows a reasonable good prediction.

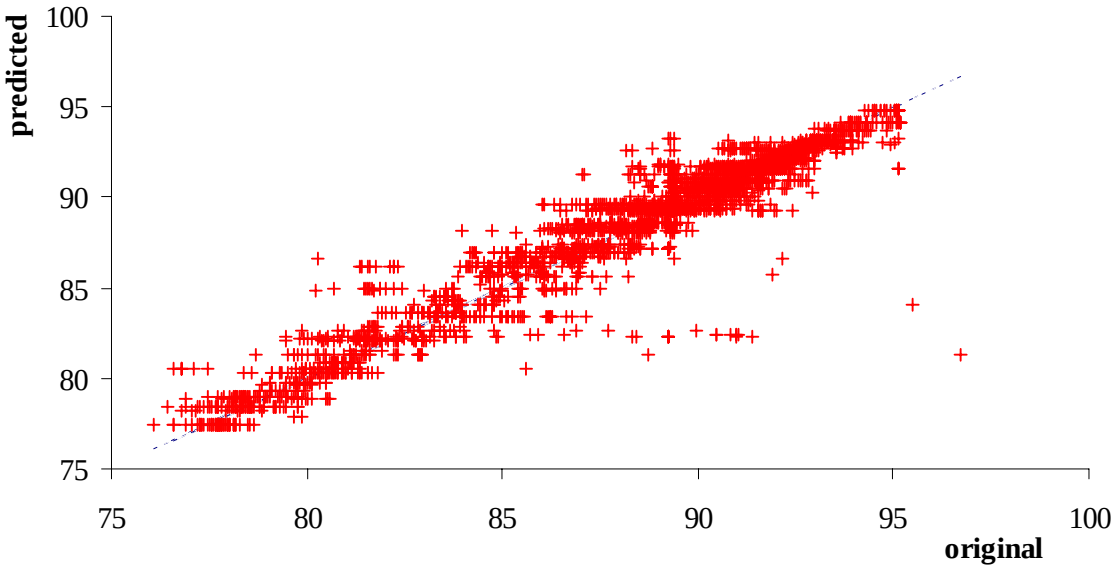


Figure D.8 : Predicted versus original values for the regression tree with a gap (MSE = 1.048 and MAE = 0.558)

	MSE	MAE
Memory depth 7	0.742	0.464
Mask with gap	1.048	0.558

Table D.6 : MSE and MAE for both regression trees[1]

From Table D.6, it can be seen that although the regression tree used here has more terminal nodes than the previous one from section 0; its predictions are slightly worse.

D.4 Conclusion

The usage of gaps in primary masks allows to go beyond the maximal computable mask. However, what happens if two entries would be important as a whole, but if they are separated by the gap? Thus, more research is needed to untangle some problems associated with this approach. This example shows that a priori knowledge has to be used to propose meaningful, albeit high in cardinality, primary masks.

[1] The number of points differs when applying the different masks. Hence, it is opt to use the mean squared error and mean absolute error.

Abbreviations

5+1NN	5+1 Nearest Neighbour methods that applies for SAPS-II
5NN	5 Nearest Neighbour method
5UNN	5 Unit-rescaled Nearest Neighbour Method
AI	Artificial Intelligence
AIC	Akaike Information Criterion
ARMA	Auto-Regressive (AR), Moving Average (MA)
C4.5	Name invented by Quinlan, no abbreviation
CART	Classification and Regression Trees
CHAID	Chi-squared Automatic Interaction Detection
DBMS	Data Base Management System
DM	Data Mining
FIR	Fuzzy Inductive Reasoning
FSA	Finite State Automaton
FST	Finite State Transducer
GSPS	General System Problem Solving
GST	General System Theory
I/O	Input/Output
ID3	Inductive Dichotomizer 3
KDD	Knowledge Discovery in Databases
LAD	Least Absolute Deviation
LS	Least Squares
MAE	Mean Absolute Error
MGC	Membership Grade Coefficient
MIMO	Multiple Input Multiple Output
MIS	Management Information System
MISO	Multiple Input Single Output
ML	Machine Learning
MSE	Mean Squared Error
NP	Nondeterministic Polynomial

OLAP	On-Line Analytical Processing
OR	Observation Ratio
SAE	Sum of Absolute Errors
SAPS	System Approach Problem Solver
SISO	Single Input Single Output
SQL	Structured Query Language
SSE	Sum of Square(d) Errors
ST	State-Transition
UC	Universal Coupling

GLOSSARY

Glossary

Note: References in the glossary contain in front the chapter number. For example: the reference [1: Klir 1969] can be found at the end of chapter 1.

Activity: The ensemble of the variations in time of all the quantities under consideration at a given resolution level will be called the activity of the system, [1: Klir 1969, p. 41]. The activity of a system is a record of data for a given time interval, or the ensemble of the variations in time of all the quantities under consideration at the given resolution level. If integers are used after abstraction of the system quantities to obtain system variables, then the activity matrix is called normalised, [1: Klir 1969, p. 115].

Akaike information criterion (AIC): A criterion that attempts to prevent overfitting by penalising each extra parameter. Used for model selection.

APL: APL is a programming language originally created by Ken Iverson in the 1960's. APL began as a notation to describe mathematical ideas. The notation consists of a set of symbols and a syntax to describe the processing of data. The power of APL comes from its direct manipulation of n-dimensional arrays of data. The APL primitives express broad ideas of data manipulation. These rich and powerful primitives can be strung together to perform in one line what would require pages in other programming languages, [from www.acm.org/sigapl/apl.htm]

Attribute focusing involves a (horizontal) data reduction in the parameter space. It creates a target data set by focusing on a subset of variables — restrict the parameters used to do the analysis. Examples: possible relevant variables are occupation, marital status, possessions, bank accounts with amount of money on them, age, other loans, etc., while irrelevant variables are hobbies, religion, race, etc. Thus, attribute focusing is mainly done manually to restrict the search space by elimination of (what are assumed) irrelevant dimensions. One could try different sets of attributes and run the whole process to see if useful patterns emerge (because some data mining algorithm can detect irrelevant variables), but this may become impossible due to computational complexity in high dimensional spaces. Hence, usually a maximum acceptable set is used and during the pattern search irrelevant attributes are eliminated (e.g., they do not contribute to the information contents).

Backward stepwise methods: see forward stepwise methods

Base model: a model, which description can never be fully known, although certain aspects of its description may be accepted as known. The base model provides a complete explanation of the behaviour of a real system; it may be expected to comprise many, many components and interactions.

Basket analysis: a method of analysing customers' behaviours and looking for patterns in their buying preferences.

Bayes error rate: In the statistical approach, an overlap between the probability distributions of the different classes (near decision surface) is an inherent aspect of the approach. A decision surface or line will be based on a threshold. The overlap has as effect that there will be always a lower bound to the error rate of classification (Bayes error rate) no matter what classification method is used. The lower bound can theoretically be reached by the optimal attainable classification rule, i.e., the Bayes' classification rule, which is defined as such. The only way to get rid of the overlap and thus the inherent minimum bound on the error rate is to import extra information.

Boosting: Boosting is a technique for generating and combining multiple classifiers to give a superior prediction accuracy. For example, when a new case is to be classified, each classifier votes for its predicted class and the votes are counted to determine the final class. Unfortunately, boosting doesn't always help.

Case (synonym: example, instance, object): see object.

Causal dynamical system: the outputs of a system at a certain time instance t_0 are not influenced by inputs at $t \geq t_0$.

Certainty measure (evidence measure): indicates the significance of a pattern. This significance is often measured by a statistical criterion. Certainty is related with accuracy. In other domains (fuzzy set theory), confidence measures are used. Related with this is also a measure of support.

Change and deviation detection discovers the most significant changes in data from previously measured values; e.g. fraud detection.

Classes: They can be defined in some different ways. Classes can be pre-defined by a partition of the sample space, i.e., as a function of the attributes themselves (unsupervised learning). Another definition is that classes correspond to labels for different populations (classification), e.g. dogs and cats are two classes and it can be said whether an animal is a cat or a dog (crisp). Finally, they can be defined as resulting from a prediction problem (a class is an outcome that must be predicted from knowledge of the attributes). The latter case will be the definition applied in the dissertation.

Clustering seeks to identify a finite set of categories or clusters to describe the data. The categories may be mutually exclusive and exhaustive or consist of a hierarchy of overlapping categories. It is the task of segmenting a heterogeneous population into a set of more homogeneous clusters. The grouping is based on self-similarity. It is often used as a first step in market segmentation.

Complex (black-box) systems: systems with many I/O variables and with a deep memory. Complex systems can also be defined in other ways, e.g., on UC-structure, but this dissertation restricts the definition to parameters that relate to mask cardinality (see chapter 1 for a definition). For a finite memory machine, it can be defined as the relevant number of arguments in the output equation.

Comprehensibility is not necessarily the same as simplicity, but often shows the same tendency. Comprehensibility is hard to formalise.

Conditional Entropy: The conditional entropy of Y given $X = i$ is given by

$$H(Y \mid X = i) = -\sum_{j=1}^{J} p_{Y|X}(j \mid X = i) \log_2 p_{Y|X}(j \mid X = i)$$

where the summation is over all possible outcomes of Y, which is set to $\{1,2, \ldots, J\}$.

The conditional entropy of Y given X is given by the average of the conditional entropy of Y given $X = i$, i.e.,

$$H(Y \mid X) = \sum_{i=1}^{I} p_X(i) H(Y \mid X = i)$$

where the summation is over all possible outcomes of X, which is set to $\{1,2, \ldots,I\}$

Confusion matrix: A confusion matrix lists the correct classification against the predicted classification for each class (cross-classification of predicted versus true class). A confusion matrix is a two dimensional matrix that contains on its diagonal elements the number of correctly classified cases and on its off-diagonal elements the number of wrongly classified cases. As accuracy is a measure for the overall correctness of the classifier, it may be defined via a confusion matrix. From a confusion matrix some new measures can be deduced, e.g. sensitivity = true positives / actual positives.

Cost matrix: If different types of errors have a different impact on the cost of misclassification, a cost matrix should be used. This is expressed by simple multiplication of an error with its misclassification cost. Usually, a correct classification costs nothing (no penalty). The multiplication factors can be represented in a misclassification cost matrix. The assignment of cost factors is still a difficult decision usually made by an expert.

Cost of classification: it may be very well the case that construction of a classifier requires such cost in obtaining a valuable training set that it is prohibitive. Examples can be found in the medical field where human guinea pigs are not ethical (and economical) justifiable. A less obvious illustration occurs when the decision to do a test is based on both the cost of test and the cost of misclassification. If a test costs more than the corresponding misclassification, then there is no point in doing it. Hence, the cost of test should be taken into account when constructing classifiers. A special case arises when tests are inexpensive, relative to the cost of classification errors. This may be the case in the medical domain, where the live of a patient is at stake. Sometimes, not all tests are necessary. Especially when tests are relatively expensive, it may be advisable to lower (total) costs by alternating tests and classification decision. Hence, do a test, examine the result and the classification error cost, and then decide if further testing is feasible or justified.

CTRL-C: an interactive matrix manipulation language developed originally for computer-aided control system design, and enhanced later for other purposes such as signal analysis and statistical operations. It is another Matlab dialect, which basically extends the Pascal programming style, operating conveniently on matrix data structures [from www.ece.arizona.edu/~cellier], and [2: SCT 1985],

Curse of dimensionality: The higher the dimensionality in a data set, the sparser and more spread the data points are, while the number of parameters needed to specify a distribution increases. If the variables are not independent, correlation has to be taken into account raising the number of parameters involved even higher.

Customer retention: All actions, which rely on knowledge of the customer, to inspire loyalty from a customer. This knowledge consists of the specific needs and preferences of the customer, which are learnt from (past) personal interactions. Large firms have too many customers to keep this up. Hence, they want to learn automatically the lifetime value of each customer to know which ones are worth investing money and effort into. They try to achieve this via noticing (in an automatic fashion) via all kinds of registration machines (e.g., bar code readers, surveys, …) what the customer does.

Data augmentation: The data is augmented with additional variables (fields) derived from existing ones. This is natural in hypothesis testing where the additional variables may arise

naturally, but also in knowledge discovery it may be advisable, for example, to define a body mass index in certain studies about overweight and other variables.

Data cleaning concerns the removal of noise, the analysis of outliers if appropriate, or trying to construct a model to account for the noise. A strategy for handling missing data has to be chosen. In data pre-processing all available a priori knowledge should be used (e.g., what about an outlier?).

Data focusing involves a (vertical) reduction in the observation space if the data set may contain too many records. It may be obtained by looking at data samples that are relevant for the problem at hand. It is not to be confused with sampling the data for computational reasons, which can be applied too. For example, only look at (previously or) existing customers who have a mortgage loan when the goal is to seek a pattern for credibility assessment. Data focusing can be done by resampling from the databases. Different sampling techniques may be presented. The alternative is to select manually a subset of data records to process in the next steps (e.g. do not select records with missing values). Data focusing is also used to reduce computing time in the successive steps of KDD. If one has a representative sample of the whole database, then the number of records to work with is only a fraction of the total number of records. Finally, clustering techniques may also help by grouping the data records and thus inducing an aggregation.

Data mining refers to a class of methods that are used in some of the steps making up the overall KDD process. A general definition of data mining and the link between data mining and KDD is given in [5: Fayyad 1996]: "*Data mining is a step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produces a particular enumeration of patterns over the data set*". This enumeration of patterns can not be done exhaustively over the entire infinite space of patterns. Computational restrictions are used to delimit this space. Thus, data mining involves fitting models to, or determining patterns from observed data. The fitted models play the role of inferred knowledge whether or not they reflect useful or interesting knowledge. The latter issue belongs to the domain of KDD. The terms 'KDD' and 'data mining' are often used interchangeable. Fayyad [5: 1996], however, makes a clear distinction between KDD and data mining: "*KDD process is the process of using data mining methods (algorithms) to extract (identify) what is deemed knowledge according to the specifications of measures and thresholds, using the database of facts along with any required pre-processing, subsampling, and transformations of the facts*". Some institutions like SAS consider the data mining process as the whole KDD process (see [5: SAS Inst. 1996], page 6). Management Information System (MIS) people also tend to call the whole KDD process a data mining process, see [5: Berry 1997], page 5: "*Data mining, as we use the term, is the exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules*". It is interesting to note that the term 'automatic' or 'semiautomatic' is used. In data mining according to SAS and Berry the goal is defined, the data is preprocessed, a model is specified, fitted, evaluated, and new knowledge is consolidated. These steps are exactly the basic steps in KDD. See [5: Berry 1997] page 5, where he writes: "*We assume that the goal of data mining is to allow a corporation to improve its marketing, sales, and customer support operations through better understanding of its customers*". The goal has to be specified out and the understanding of its customers is the consolidation of the newly gained knowledge. The term 'data mining' has been mainly used by statisticians[1], data analysts and the community. KDD has been used more by AI (Artificial Intelligence) and the

[1] These boundaries are not that crisp

ML (Machine Learning) people (recently, there is a Special Interest Group in KDD, which is abbreviated by SIGKDD). As KDD systems typically draw upon methods, algorithms, and techniques from diverse fields such as machine learning, statistics, AI, exploratory data analysis, etc., it is not unusual to find data mining methods from these fields.

Data projection: Dimensionality reduction or transformation methods that further reduce the number of variables under consideration. Two examples are principal component analysis and Fourier analysis.

Data set: a set of facts e.g., cases in a database.

Data warehouse: Data coming from many sources (billing records, scanning data, registration forms, etc.) are gathered together and organised in a consistent and useful way (common format) in a so called data warehouse. The latter plays the role of enterprise memory. These gathered and merged data is still not information, thus another step consists of analysing and understanding the data, and to turn it into actionable information. This is the KDD process, which plays the role of enterprise intelligence. Data warehouses are relatively new (since the 90's) and the term data warehousing has become very popular in MIS environments. They consider a data warehouse as a suitable repository that contains data that is analysed for business decisions. It stands for collecting and cleaning transactional data and preparing it for online retrieval for decision support (typical report generation or OLAP). Its function is to handle missing data, to keep historical data (which operational databases do not maintain), and to consolidate data (aggregation, summarisation) from heterogeneous sources, which typically use inconsistent data representations, codes and formats which have to be reconciled. Data warehouses cannot be bought off the shell because each company is different and has its own specific requirements, end-user needs, etc. It is clear that the database community has a large hand in this area of research. Notice that a data warehouse is also much different from a production database: a data warehouse stores historical data. Furthermore, the structure of a data warehouse is not static. Common database techniques such as normalisation are not very well applicable anymore. This is tight related to OLAP, where the performance of queries is of the uttermost importance. Hence, the design of a good data warehouse is an art on itself. It is not surprisingly to see that a whole field with regard to data warehousing has emerged (with its own terminology).

Data warehousing is the process of bringing together data from throughout an organisation for decision-support purposes. The latter term should be taken very generally, i.e., gathering reports, OLAP, and data mining. Data brought together is dirty: there are semantic and syntactic conflicts; there are missing values, wrong values, etc. The data warehouse facilitates the data mining efforts by doing a low-level data pre-processing of such data. Low level cleaning of data happens in the data warehouse. Some actions are amalgamation of operational data from different sources, checking of redundancy, removal of duplicates, correction of typo's, indication of missing values, verification of data and comparison of scales. Semantically equivalent values should preferable be expressed by syntactically equivalent values (same data model). Detecting data anomalies and rectifying them early in the data warehouse has huge payoffs in the whole process, but it still takes up from 60 - 70 % of the total effort of virtuous cycle, [5: Han 1999]. Hence, this practical and low-level data cleaning step should happen always and independent of any goal setting. All data in the database should be consistent, verified and use a same data model. Furthermore, a first kind of data reduction can already happen in the data warehouse (data cube aggregation) by eliminating unwanted resolution (e.g., precision for numbers, address details for addresses, string compression).

Data-suitability problems: Data-suitability can best be explained with some examples. If, for example, one wants to analyse buying habits of customers in order to market differentially

to new customers and to long-standing customers then one needs data over a sufficient long period. If a database is purged every six-months or year, then long-standing customers who have not bought anything during the last six months or year will be overlooked. Another example can be found in fraud detection. It is unlikely to have a data set with known fraudulent cases from which one may extract useful patterns for fraud detection. Only deviation from normal patterns can be used to suspect fraud.

Decision surfaces: Hypersurfaces that separate the measurement space in regions that correspond with classes. Moving across a decision surface changes the decision taken.

Dependency modelling finds a model that describes significant dependencies between variables. At the structural level, it is specified which variables are locally dependent on which other ones. At the quantitative level, the strength of the dependencies is indicated. This is what SAPS tries to do.

Design set (synonyms: training set, learning set): A data set used for inducing a model.

Directed knowledge discovery: In directed knowledge discovery, the task is to explain the value of some particular field (target, response, and dependent variable) in terms of all the other fields (input, explanatory, and independent variables or stimuli).

Dynamic discretisation (KDD): Quantisation happens during the model induction. Tree classifiers do dynamic discretisation. Discretisation in KDD is what is quantisation in GST (although it is also called discretisation in certain fields).

Eager evaluation/classifier: Contrary to lazy learning methods, eager methods construct a model beforehand for classification. A parsimonious classifier is built that allows very fast classification on a test set. All necessary knowledge for classification is built in during the construction of the classifier. It takes the classifier more time to learn compared to a lazy classifier, but the classification is much faster. Examples of eager classifiers are neural networks, classification trees, regression trees, etc.

Expressiveness: measure of the information content in a variable, related with resolution.

Finite automaton (deterministic version) (DFA): A DFA is defined as a quintuple $< \Sigma, S, s_0, \delta, F >$, where
(1) Σ is the input alphabet (a finite nonempty set of symbols),
(2) S is a finite nonempty set of states
(3) s_0 is the start (or initial) state, $s_0 \in S$,
(4) δ is the state transition function; $\delta: S \times \Sigma \rightarrow S$
(5) F is the set of final (or accepting) states, $F \subseteq S$ (possibly empty).
from [3: Carroll and Long 1989, p. 30]. See also non-deterministic finite automaton.

Finite-memory machine (FMM): A FMM is a discrete system with a specified finite set of stimuli and responses and a given function

$$y_i = h_i(u_i, u_{i-1}, \cdots, u_{i-a}, y_{i-1}, y_{i-2}, \cdots y_{i-b})$$

where $u_i, u_{i-1}, \cdots, u_{i-a}$ are stimuli at time instances $t_i, t_{i-1}, \cdots t_{i-a}$, respectively, and $y_{i-1}, y_{i-2}, \cdots y_{i-b}$ are responses at time instances $t_i, t_{i-1}, \cdots t_{i-b}$, respectively ($i$ = 0, 1, …). A probabilistic finite-memory automaton (PFMM) can be defined by assigning several values of y_i with different probabilities to a single set of values given by $u_i, u_{i-1}, \cdots, u_{i-a}, y_{i-1}, y_{i-2}, \cdots y_{i-b}$.

Forward stepwise methods: Forwards stepping makes the model more complex by adding an extra variable, while backward stepping simplifies the model by throwing out variables that are least relevant. Both methods do not necessary lead to the same end-model for both represent a different limited search in the search space.

Generality determines the fraction of the population a pattern refers to.

Good mental fit: Typical the aim of ML. It can be used as a definition of ML. As described in [5: Michie et al. 1994]: ML is interested primarily in models that are "*simple enough to be understood easily by humans*", i.e., in models that give a good mental fit. The latter term is also another expression for the high degree of comprehensibility (and thus acceptability) with the cognitive model of an expert.

Greyness spectrum: White-box and black-box systems are situated at the end of a spectrum of intermediate system colours. This results in shades of grey that indicate (qualitatively) how much a priori knowledge is available. This situation is nicely summarised by Karplus [1: 1976] in a spectrum of modelling, which is depicted in chapter 8. Remark the shift from quantitative to qualitative when going more into the darker regions of the spectrum.

Hypothesis testing is a top-down approach. It consists of substantiation or disapproval of preconceived ideas. The goal is to confirm the correctness of a priori knowledge[2], e.g., are young people more likely to respond to a given offer? A hypothesis is a proposed explanation whose validity can be tested. Hypothesis testing is popular in the statistical society. A hypothesis is a formalisation of an idea or mental model, which usually follows from a clear statement of a problem. In ML: a hypothesis is a candidate representation of a target mapping.

Indicator function: The indicator function is denoted in general by $I_{(statement)}$. It is defined to be 1 if the statement inside the parentheses is true, otherwise it is zero. E.g.

$$I_{(x \in A)} = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Induction by enumeration states that, after observing that something is the case quite some number of times, it is always the case. It is the one but worst scientific method of induction (even worse is not looking at all at the data and take a default decision).

Inductive bias: Formally, for a learning algorithm L, instances X, a target concept c, and training examples $D = \{\langle x, c(x) \rangle\}$, let $L(x_i, D)$ denote the classification for x_i learning after training on D, then the inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D, $(\forall x_i \in X)((B \wedge D \wedge x_i) \succ L(x_i, D))$, where $P \succ Q$ stands for P logically entails Q, or Q is inductively inferred from P. One may regard the inductive bias as a piece a priori knowledge, so that the classification rule can be considered as deduced from B and D. This corresponds with an inductive leap.

Inductive learning hypothesis: "*Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples*" [5: Mitchell 1997].

Knowledge Discovery in Databases (KDD): Both Fayyad [5: 1996] and Berry [5: 1997] give definitions. They all come down to the same thing, except for the degree of automation involved in the KDD process (see data mining) Hence, the definition of KDD taken here is, [5: Fayyad 1996, page 6]: "*Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data* (from large databases[3])"

---

[2] More strictly said: one tries to falsify a hypothesis (cf. Popper).

[3] Added by the author

Knowledge discovery is a bottom-up approach. It starts with data and tries to get previous unknown knowledge, e.g., which target group of customer is more likely to buy our product, what kind of customers group (clusters) are there? The goal is to discover useful relationships or clusters in the data. It tries to find patterns without a predetermined idea or hypothesis about what the pattern may be. The machine learning society has since long been interested in knowledge discovery.

Lazy learning/evaluation/classification: Little time is needed for the construction of the classifier, but the classification on a test set takes more time than an eager classifier. Most of the necessary knowledge for classification still resides in the data set, which is used during classification. Example-based methods are sometimes referred to as 'lazy' learning methods because they delay processing until a new instance pops up to for classification. This can be contrasted to 'eager' learning methods. Another example is classification by enumeration.

Learning set (synonyms: Design set, training set): see design set.

Lumped model: Due to the complexity of a base model, a relatively more practical model has to be found under a given set of conditions (experimental frame), [1: Zeigler 1976, chapter 2].

Machine learning: A definition of can be found in [5: Mitchell 1997]. Mitchell states that "*a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*". The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience, i.e., computer programs that learn. Machine learning attempts to answer questions such as

(1) How does learning performance vary with the number of training examples presented?
(2) Which learning algorithms are most appropriate for various types of learning tasks?
(3) What algorithms exist, which is best, convergence for sufficient training data, how much data points to have sufficient data?
(4) How to include a priori knowledge?

One has to determine exactly what type of knowledge will be learned and how it will be used by the performance program. Usually this boils down to searching in a vast search space of choices to the best solution for the learning problem at hand, but for which the best search strategy is not known (optimisation problem).
In addition, the concept of a good mental fit is of paramount importance in ML.
In Knowledge Discovery, machine learning is most commonly used to mean the application of induction algorithms, which is one step in the knowledge discovery process. This is similar to the definition of empirical learning or inductive learning in Readings in Machine Learning by Shavlikand Dietterich. Note that in their definition, training examples are ``externally supplied," whereas here they are assumed to be supplied by a previous stage of the knowledge discovery process. Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to "learn".

Maximal allowable mask: a special kind of primary mask for which the cardinality is maximal. Informally, except for the output entry (+1 in the mask), all other entries are -1. This is contrary to a primary mask, which may have some entries set to zero (thus a lower cardinality for a given number of variables and mask depth).

Mealy machine or finite state-transducer (FST): A FST is defined by a sextuple $< \Sigma, \Gamma, S, s_0, \delta, \omega >$, where the specifications are as for a DFA, but with

(5') Γ denotes the output alphabet
(6) ω denotes the output function; $\omega: S \times \Sigma \rightarrow \Gamma$

This corresponds with what Klir [1: 1969] referred as a finite state machine (terminology is not always consistent when developed in different domains).

Measurement space (synonyms: feature space, instance space): (multivariate) space spanned by the variables. Each vector of measurements for a particular object corresponds to a point in the measurement space.

Missing values: One has a missing value when the value for a variable is unknown or does not exist. For example, absence of a co-ordinate in a measurement vector.

Model structure: it may have different meanings, but the one in this dissertation defines it as the type of the system description: e.g., differential equation form, etc.

Moore machine: A Moore machine with a distinguished start state is a sextuple $< \Sigma, \Gamma, S, s_0, \delta, \omega >$ defined in the same way as a Mealy machine, but where
(6') ω denotes the output function; $\omega: S \rightarrow \Gamma$
Non-deterministic variants of a Mealy or Moore machine could be defined by relaxing the determinism of the state-transition function and the output function. One has then for
- the state transition function: $\delta: S \times \Sigma \rightarrow \wp(S)$
- the output function: $\omega: S \times \Sigma \rightarrow \wp(\Gamma)$ (Mealy) or $\omega: S \rightarrow \wp(\Gamma)$ (Moore)
In this way one can define a non-deterministic Mealy and non-deterministic Moore machine.

Mutual information: The mutual information between two random variables *X* and *Y* is defined by $H(Y;X) = H(Y) - H(Y|X)$. It measures the average reduction in uncertainty about *y* that results from learning the value of *x*., or vice versa. A fundamental inequality first shown by Shannon states that $H(Y) \geq H(Y|X)$, thus the mutual information is always positive. It is also symmetrical, i.e., $H(Y;X) = H(X;Y)$.

Noiseless cases (or noiseless situation): An ideal situation for learning is where every measurement vector in the measurement space exactly belongs to one and only one class. In that case, the classes are said to be perfectly separable and the situation is called noiseless or degenerate. The classification problem encompasses then the generalisation of the known class memberships of the samples in the design set to other points. The noiseless approach is very typical for the machine learning domain.

Non-deterministic finite automaton (NDFA): A NDFA is a quintuple $< \Sigma, S, S_0, \delta, F >$, where the specifications are the same as for a DFA, but
(3') S_0 is the set of start (or initial) states, $S_0 \subseteq S$,
(4') $\delta: S \times \Sigma \rightarrow \wp(S)$ ($\wp(S)$ is the power set of *S*)
from [3: Carroll and Long 1989, p. 119].

Non-homogeneity can be interpreted in several ways:
(1) A mixture of data types (categorical or interval).
(2) A standard or non-standard data structure.
(3) Non-homogeneity in its strict meaning indicates that different relationships hold between variables in different parts of the measurement space.

Novelty: implies the (often unexpected) deviation of a pattern from prior knowledge.

Nuggets problem type: This is a problem type described in [5: Klösgen 1996], which is prominent when data is highly inconclusive or when the distribution of cases to classes is far from uniform. Some classes may have almost no representants. Depending on the goal this may or may not be a problem (e.g., taking classes together that are not interesting for the goal).

Off-line learning: A training set is offered as a whole.

OLAP: OLAP (On Line Analytical Processing) is a multi-dimensional data analysis that computes summaries and breakdowns along many dimensions. OLAP is superior and better than the SQL (Structured Query Language) for analysis of data. The data is usually stored in a data cube, which shows three chosen dimensions of the data. A specific terminology is then used to summarise data (roll up), to go from higher level summary to lower level summary or detailed data (roll down, drill down, drill through), to select and project (slice and dice), etc.

On-line learning: A training set is offered on a pair by pair basis. The passing in of a particular training pair is called a presentation.

Parameter space: support space

Pattern: an expression in a certain formal language describing facts in a subset of a data set. A pattern should be simpler than the enumeration of all facts in the subset (parsimonious).

Period sampling: the sampling instants are equally spaced in time, [2: Åström and Wittenmark 1997].

Permanent behaviour: intrinsic behaviour. This kind of behaviour stems from an absolute relation, i.e., a relation valid for any activity and for all time (all modes of the system). In practice, this behaviour is difficult to distinguish from a relatively permanent behaviour (a specific mode (activity) of the system).

Physical systems: In physical systems, the quantities are measurable, otherwise, one has abstract systems.

Predictiveness: measure of the forecasting power (usually measured by accuracy).

Primary mask: called candidate mask in SAPS-II. The mask search will only generated submasks from this primary mask for evaluation and comparison. It limits the search space (not to be confused with a maximal allowable mask). The cardinality of a primary mask can be less than a maximal allowable mask, because a primary mask may be based on a priori knowledge. Examples are found in Appendix A (some entries are set zero beforehand), Appendix C (after feedback from another method) and Appendix D (a mask with a gap).

Probability density estimation consists of techniques for estimating the joint multivariate probability density function. It can be used in conjunction with classification techniques.

Problem of induction: When a general rule (no matter how complex) is learnt from observations, e.g., via induction by enumeration. It may very well be that the next observation refutes the general rule, even after observing a same case many times until then. This is known as the problem of induction. It implies that there is no secure foundation for inductive generalisations in an infinite universe.

Quantisation: mapping of sampled continuous values to a finite set of qualitative values. This is similar to digitalisation of a signal.

Raw data: real data, i.e., measured data (attributes) coming from the system

Redundancy has to do with the similarity of patterns with other (already found) ones and to what degree one pattern follows from another.

Relatively closed: the paths over which the environment acts on the system — the inputs or stimuli — as well as the paths over which the system acts on the environment —the outputs or responses — are accurately defined, [1: Klír and Valach 1967].

Richness of structure: the variability in the data should be high enough and some data attributes must capture relevant knowledge.

Robustness of a classifier: how well can a classifier cope with noise and missing values.

Sampling theorem by Whittaker in 1915 and described in [1: Klir 1969, p 71] states that "*Every continuous function of time which has a frequency spectrum with an upper frequency limit f_{max} can be accurately substituted by a finite number of its values recorded at time intervals of* $t = \frac{1}{2 f_{\max}}$" This theorem is also known as the Nyquist theorem. Later it was explored in more depth by Shannon in 1949, [2: Åström and Wittenmark 1997].

Sampling: (in the context of dynamical systems) the determination of the (optimal time) step in the case of continuous support. [2: Åström and Wittenmark 1997] gives a definition in the context of control and communication: sampling means that a continuous-time signal is replaced by a sequence of numbers, which represents the values of the signal at certain times. The inverse process is called signal reconstruction.

Selectivity condition, [1: Karplus 1976]: The selectivity condition can best be illustrated with a simple example. When considering a simple resistor we may use Ohm's law, but we know this is only a linear approximation of a more complex behaviour. Going into more detail, one could opt to look at heat effects too and chemical composition. Even size effects may play a role when the dimensions are very small (like in IC's). Besides these static considerations, one may look at the time variant behaviour of the resistor concerning the just mentioned properties (e.g., ageing effects, burning out of the resistor). Thus, when one considers a reasonably complex system that consists of many components (complex structure), one clearly has to limit the investigation of the system to the observation of certain attributes (values of certain quantities characteristic for the system). This is related to the relative closedness of a system.

Separability: The separation between the system under investigation and the environment induces boundaries, which may be hard to define, [1: Karplus 1976].

Simplicity refers to the syntactical complexity of a pattern.

Smalltalk: an object-oriented programming environment. It is exemplary for object-oriented thinking. Many dialects are available, some of them are for free, see also www.bsug.org

Soften threshold: Implemented in See5© by breaking each threshold into three ranges. If the attribute value in question is in the outer ranges, classification is carried out using the single branch corresponding to the `<=' or '>' result respectively. If the value lies in the middle range, both branches of the tree are investigated and the results combined probabilistically.

Space-time resolution level: Accompanying the space-time specification we need the space-time resolution level, which concerns the level of detail in measuring the attributes (e.g. accuracy and sampling rate/frequency of measuring).

Space-time specification: When observing what are deemed relevant attributes of a system, one must also take into account the time frame of the observations (beginning and end of measurement, sampling,…). The previous specifications that apply for a system under investigation is what Klir [1: 1969] calls the space-time specification.

Standard structure data: In a standard structure, the dimensionality of the measurement vectors is fixed. For tree classifiers, this implies that the class of questions for splitting in a tree can be standardised. It is assumed that the measurement vectors have the form $\vec{x} = (x_1, \cdots, x_M)$ with *M* fixed and each split depends on the value of only one variable.

Static discretisation: E.g., continuous values can be quantised by the use of an equal-width partitioning (uniform grid) or via equal-depth (frequency) partitioning [5: Han 1999].

Summarisation results in a compact description for a subset of data, e.g. mean, range, median, variance, etc. for all field (summary statistics, report generation with corresponding graphs, …).

Summary and Description problem type: Problem type that emphasises gaining insight: comprehensibility is important and the level of generalisation should be high enough [5: Klösgen 1996]. Few classes are often preferred, which can be obtained by attribute focusing.

Supervised learning (ML): A process in which the learner searches in the hypothesis space a hypothesis that agrees with all the examples in the training set. In KDD, this is called directed knowledge discovery approach.

System: In the investigation of 'something', we have to confine our research to a certain part of nature (the universe) that interests us. Mesarović [1: 1969] formalises a system *S* as a set of pairs of sequences of input and output symbols generated by an automaton by the application of *P* and *G* on a given input sequence. The automaton is defined as a quintuple (*A*, *B*, *Z*, *P*, *G*) such that *A* is the set of input symbols, *B* is the set of output symbols, *Z* is the set of states, *P* is the next state function, i.e. $P: A \times Z \to Z$, *G* is the output function, i.e. $G: Z \times A \to B$ Zeigler states in [1: Zeigler 1976] that a system consists of a static structure (a time base, input stimuli, output responses, and a state set (memory of the system)) and a dynamic structure (the functions between outputs and inputs)

Target mapping: In supervised learning, computational learning aims to produce an implementation of a mapping between two sets of objects. This is called the target mapping. Hence, computational learning is a process in which a learner produces a representation of a target mapping working from training information derived from some environment [5: Thornton 1992]. Based on a training set a target mapping must be constructed that enables correct outputs to be returned for inputs that do not appear in the training set. However, depending on the learning algorithm the constructed representation will vary. The concept of a target mapping allows for definitions for inductive learning hypothesis, connectionist learning, etc.

Target marketing: In advertising, an advertiser wants to reach a certain target public (prospective customers). Target marketing involves the identification of the variables that are important in identifying these customers with a higher responsiveness.

Test set: Set on which the validation of the induced model happens, also called validation set.

The virtuous circle of data mining is described by a cycle that consists of [5: Berry 1997]:
(1) identifying the business problem. This results in setting the goal in KDD module.
(2) using data mining techniques to transform the data into actionable information. This is the remainder of the KDD process module.
(3) acting on the information. This action is not considered in the KDD.
(4) measuring the results and put them in the database for possible further processing. This action is not considered very explicitly in the KDD, i.e., one may consider two kinds of feedback from the results to the goal.

    (a) does it satisfy the goal?
    (b) will the action satisfy the goal (is the action wrong, does the goal need adjustments)?

Only item (a) is considered in KDD. Notice that item (1) and (3) are pure enterprise issues, which are taught in management courses. The module 'measure results of action' thus refers specifically to measures of business value that go beyond response rates and costs, beyond average and standard deviations. It is more than just knowledge consolidation; it is looking at

the real world effect of applying the information. Based on the results of the actions undertaken, one can define new goals or discover new business problems that have to be investigated further. The virtuous cycle can also be applied to scientific discovery, where found information can often be tested by experiments (act on the information) and where as a result new goals (lines of research) can be set forward.

Training set (synonyms: Design set, learning set): see design set.

Undirected knowledge discovery: In undirected knowledge discovery, there is no target field. This is contrary to directed knowledge discovery.

Unsupervised learning/classification refers to the process of defining classes of objects: the goal is to formulate a class structure (deciding how many classes and the assignment of the objects to the classes). A typical example is clustering analysis. In KDD, this is called undirected knowledge discovery [5: Berry 1997].

Usefulness: relates a pattern to the goal set forward.

DUTCH SUMMARY

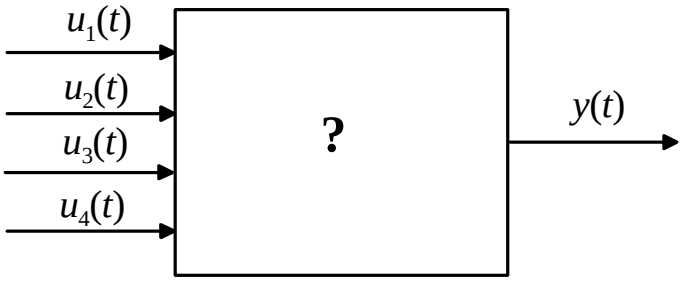
Nederlandstalige Samenvatting

Dit doctoraatswerk levert twee fundamentele bijdragen voor een op patroonherkenning gebaseerde inductie van voorspellingsmodellen voor dynamische tijdsinvariante 'black-box' systemen. De eerste bijdrage betreft een methodologisch-theoretische uitbreiding van de mogelijkheden van een experimenteel software pakket, SAPS-II genoemd, die niet-parametrische systeemidentificatie van dynamische systemen beoogt. SAPS-II is ontworpen door Cellier [2: 1991][4] en zijn team. De andere bijdrage betreft het opzetten van een veelbelovende link van de onderliggende methodologie van SAPS-II met een steeds populairder wordend wetenschappelijk domein dat, abusievelijk, 'data-mining' wordt genoemd. Bovenstaande link heeft tot gevolg dat een arsenaal van additionele methoden beschikbaar komt voor SAPS-II.

1 Initiële probleemstelling

De motivatie om vanuit SAPS-II te starten, had te maken met een fundamenteel probleem in verband met de complexiteit van de te identificeren 'black-box' dynamische systemen. De term 'black-box' verwijst naar het ontbreken van a priori kennis over het systeem. Voor zo een type systemen zijn enkel de in- en uitvoer waarden bekend als functie van de tijd (Figuur 1).

Het gedrag van een systeem kan gemodelleerd worden via patronen. Elk patroon staat voor een bepaalde afhankelijkheidsrelatie tussen een uitvoer en de andere in- en uitvoeren.



Figuur 1 : Een 'black-box' systeem met 4 invoeren en 1 uitvoer

Voor het systeem in Figuur 1 zou het volgende patroon kunnen gelden (hierbij stelt $\tilde{f}$ een niet nader gespecificeerde afhankelijkheid voor):

$$y(t) = \tilde{f}(u_1(t-3\Delta t), u_3(t-3\Delta t), u_2(t-\Delta t), u_3(t)) \qquad (1)$$

Bovenstaand patroon stelt dat een uitvoer *y* op een tijdstip *t* afhankelijk is van

- een invoer u_1 op 3 tijdstappen terug ten opzichte van het beschouwde tijdstip *t*,

[4] Omdat de referenties in het doctoraat per hoofdstuk staan, wordt het hoofdstuknummer hier ook aangeven in de referentieverwijzingen. Eigen publicaties vindt men in de Engelstalige introductie.

- een invoer u_3 op 3 tijdstappen terug ten opzichte van het beschouwde tijdstip *t*,
- een invoer u_2 op 1 tijdstap terug ten opzichte van het beschouwde tijdstip *t*,
- een invoer u_3 op hetzelfde beschouwde tijdstip *t*.

De uitvoer *y* is niet afhankelijk van de invoer u_4 in dit patroon.

Met een 'complex systeem' wordt dan een systeem bedoeld met een groot aantal in- en uitvoeren en/of waarvoor het afhankelijkheidspatroon diep terug gaat in de tijd. Dit stemt overeen met een groot aantal argumenten voor $\tilde{f}$ in vergelijking (1). Bij het patroon in vergelijking (1) werd geen enkele specifieke wiskundige structuur verondersteld (een lineaire vergelijking, een differentiaalvergelijking, e.d.), waarvoor bepaalde parameters dienen te worden geschat. In dit verband spreekt men, in de systeemtheorie althans, van een niet-parametrische benadering[5]. Tenslotte wordt verder verondersteld dat het patroon in vergelijking (1) niet verandert in de tijd, m.a.w. het is tijdsinvariant. Dit betekent dat de inwendige structuur en/of de intrinsieke parameters van het systeem niet tijdsafhankelijk zijn, of dat observaties in ieder geval geen drift in de tijd vertonen (stationariteit) gedurende de tijdspanne waarin de systeemidentificatie-benadering van toepassing is (opstellen *en* gebruik van het model).

Men kan nu de probleemstelling die ten grondslag ligt aan het totstandkomen van dit doctoraat als volgt omschrijven: *SAPS-II was niet in staat om zeer complexe (systeemgedrag) patronen te identificeren*.

2 Doelstelling van het doctoraat

Uit het voorgaande mag men stellen dat dit doctoraat een methode zoekt die de mogelijkheid biedt tot een niet-parametrische, op patroonherkenning gebaseerde, systeemidentificatie van *complexe realistische* dynamische tijdsinvariante 'black-box' systemen. Deze thesis beoogt de inherente beperkingen van SAPS-II, wat betreft de identificatie van *complexe* systemen, drastisch te verminderen. Dit was mogelijk door de probleemstelling vanuit verschillende invalshoeken te benaderen en vervolgens twee fundamentele wijzigingen door te voeren in de onderliggende methodiek.

De eerste wijziging situeert zich in het domein van algemene systeemtheorie (General System Theory), afgekort door GST. De oplossing in dit domein betreft een verandering van zoekstrategie. Deze wijziging vormt het onderwerp van hoofdstuk 4, waar ook een rechtvaardiging voor de toepassing van de nieuwe methode wordt gegeven.

De tweede wijziging is veel ingrijpender. Hierbij wordt de initiële probleemstelling getransformeerd naar een equivalente probleemstelling in het domein van kennisextractie in gegevensbanken (Knowledge Discovery in Databases), afgekort door KDD. Dikwijls wordt de term KDD vervangen door de meer populaire (maar onjuiste substitutie) term 'data-mining[6]'. De oplossing in het 'data-mining' domein is van fundamentele aard. Zij grijpt terug naar de principes die aan de basis liggen van SAPS, en die terug te vinden zijn in GSPS (General System Problem Solving) [1: Klir 1985]. Dit heeft eveneens als gevolg dat alle verdere modelleeracties nog maar weinig te maken hebben met de oorspronkelijke benadering in het systeemtheoretisch domein (hoofdstuk 7). De transformatie zorgt ervoor dat een verscheidenheid aan nieuwe methoden kunnen worden toegepast, die elk op zich het

[5] In statistiek slaat 'niet-parametrisch' op het al dan niet aanwezig zijn van distribuele veronderstellingen

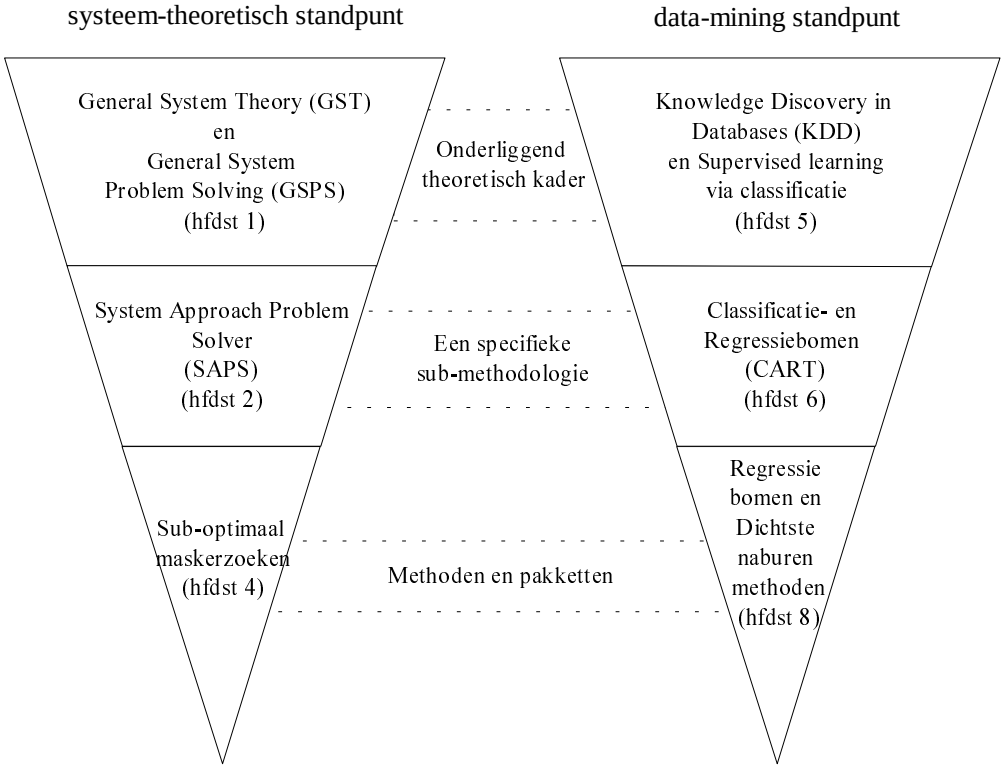
[6] Data-mining is een proces in de KDD levenscyclus (hoofdstuk 5)

onderwerp kunnen vormen van diepgaand onderzoek. Daarom is er in hoofdstuk 8 van dit doctoraat één representatieve en populaire methode uitgekozen om de nieuwe benadering te illustreren.

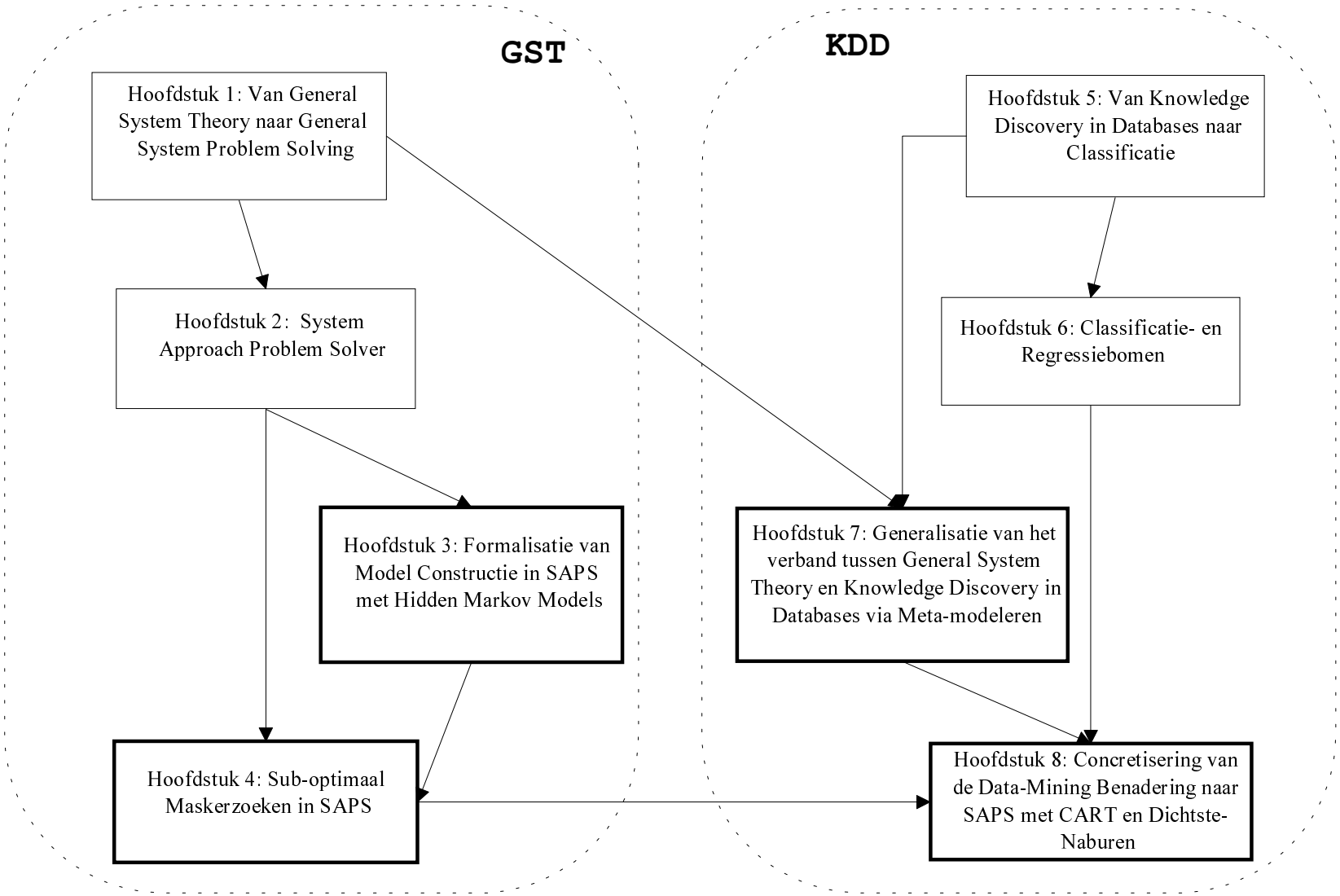
Deze doctoraatsthesis overbrugt derhalve een kloof tussen twee belangrijke wetenschappelijke domeinen, namelijk General System Theory en Knowledge Discovery in Databases. GST behandelt dynamische systemen met de bijbehorende modelconstructie-problematiek. KDD probeert bruikbare en niet-triviale patronen te herkennen in observaties, of vooropgestelde hypotheses via diezelfde observaties te bevestigen. Bijna alle applicaties in KDD die dit bewerkstelligen werken met statische gegevens. Tijdsafhankelijkheid wordt, behalve in tijdsreeksen, op een 'statische' manier behandeld. Op het eerste gezicht is een verband tussen beide domeinen voor de identificatie van *dynamische* systemen niet vanzelfsprekend. Het is dankzij het gebruik van een basistransformatie uit GSPS, in combinatie met de ingevoerde topdown benadering uit hoofdstuk 4, dat de mogelijkheid ontstaat om data-mining methoden uit het KDD domein toe te passen op SAPS.

3 Opbouw van het doctoraat

Dit doctoraatswerk wordt ingeleid met een overzicht van de twee wetenschappelijke domeinen waarin het onderzoek zich voltrekt. Het verband tussen deze twee domeinen wordt gradueel verfijnd naar het einde van de thesis toe (Figuur 2). Het doctoraatswerk bestaat dus logischerwijze uit twee hoofddelen. Het eerste deel past volledig in het domein van de algemene systeemtheorie (GST), het tweede deel in het domein van kennisextractie uit gegevensbanken (KDD) waarbij het verband tussen KDD en GST eveneens aan bod komt. Deze dualiteit wordt afgebeeld in Figuur 3. In de figuur staan de aparte hoofdstukken vermeld. Er zijn acht hoofdstukken in totaal, introductie en eindconclusie niet meegerekend. Een hoofdstuk dat in essentie een eigen wetenschappelijke bijdrage van de auteur behelst, is vet omkaderd in Figuur 3. Merk op dat deze eigen bijdragen zich situeren in beide domeinen.



Figuur 2 : De graduele focus voor beide wetenschappelijke domeinen



Figuur 3 : Overzicht van het doctoraatswerk (vet omlijnde rechthoeken zijn originele bijdragen)

3.1 Deel I: General System Theory (GST)

Deel I van het doctoraat bestaat uit vier hoofdstukken (zie Figuur 2), die zich gaandeweg meer focusseren (zie Figuur 1) op de bouw van een nieuwe applicatie die de beoogde doelstelling uit sectie 2 kan waarmaken. Uiteraard begint dit eerste deel met een overzicht van het wetenschappelijk domein GST, waarna de aandacht wordt verlegd naar het subdomein van GSPS. In hoofdstuk 2 wordt de bestaande SAPS applicatie beschreven. Hoofdstuk 3 formaliseert de constructie van een voorspellingsmodel en bakent de probleemstelling duidelijk af. Hoofdstuk 4 introduceert tenslotte de nieuwe suboptimale benadering van de patroonherkenningsinductie, rechtvaardigt deze, en illustreert deze met enkele voorbeelden. De eerste vier hoofdstukken worden in het navolgende meer in detail overlopen.

Hoofdstuk 1 brengt het theoretisch kader aan waarin de bestaande implementatie zich situeert. In dit hoofdstuk wordt een overzicht van het domein van GST gegeven en wordt de noodzakelijke terminologie verklaard. Er wordt naar een evenwicht gestreefd tussen een strikt formele en een meer beschrijvende benadering. Een begin van focussering is gemaakt door de klemtoon te leggen op black-box systemen. De inductieve benadering voor modelconstructie wordt kort uitgelegd en gecontrasteerd met de deductieve benadering. Van inductief modelleren naar systeemidentificatie is slechts een kleine stap. Systeemidentificatie bestaat meestal uit twee grote delen. Het eerste deel is structuuridentificatie. Bij parametrische systeemidentificatie volgt ook nog een tweede deel, namelijk parameterschatting. In deze thesis wordt deze benadering echter niet gevolgd. Er wordt wel een specifieke niet-parametrische benadering vooropgesteld die terug te vinden is onder de naam GSPS (General System Problem Solving). Deze methodologie is gebaseerd op patroonherkenning. Het tweede deel van hoofdstuk 1 gaat vervolgens dieper in op GSPS [1: Klir 1985]. GSPS wordt meer gedetailleerd beschreven en de nodige formalisaties worden ingevoerd. De epistemologische onderliggende gelaagde structuur van GSPS wordt uitgelegd waarbij de nadruk op de eerste drie lagen gelegd wordt. Deze zullen later essentieel blijken te zijn voor de KDD benadering. De epistemologische gelaagde benadering geeft de mogelijkheid tot een stapsgewijze inductieve (en ook deductieve) benadering van systeemproblemen. In de inductieve benadering worden de gegevens getransformeerd naar een vorm waarin tijdsinvariante patronen tussen de geobserveerde signalen op een eenvoudige manier tot uitdrukking komen [1: Klir 1969]. Het belangrijk begrip masker wordt hier voor het eerst ten sprake gebracht. De uitwijding over GSPS omvat formele concepten die in SAPS terugkomen.

In hoofdstuk 2 wordt een populaire versie van SAPS, die ontworpen is door Cellier en vervolgens de naam SAPS-II kreeg, uitgebreid beschreven, [2: Cellier 1991]. De basisbegrippen die de grondslag vormen voor de implementatie van SAPS-II worden in meer detail besproken en geïllustreerd met voorbeelden. Concepten die belangrijk zijn voor de rest van dit doctoraatswerk, zoals ‘recoding’, ‘optimal mask search’, ‘quality measures’, etc. worden besproken en geformaliseerd. Het belangrijke begrip ‘masker’ wordt nu concreet voorgesteld. Een masker is namelijk gewoon een andere representatie van een tijdsinvariant patroon in de vorm van een matrix. Het masker dat overeenstemt met vergelijking (1) is te vinden in Tabel 1. Dit masker heeft een geheugendiepte van 3 tijdstappen en een cardinaliteit (dit is het aantal niet-nul elementen) van 5. De kolommen corresponderen met de veranderlijken van het systeem (in- en uitvoeren). De rijen stellen de relatieve tijdsas voor t.o.v. de laatste rij die als referentietijdstip dient.

	u_1	u_2	u_3	u_4	y
$t-3\Delta t$	-1	0	-1	0	0
$t-2\Delta t$	0	0	0	0	0
$t-\Delta t$	0	-1	0	0	0
t	0	0	-1	0	1

Tabel 1 : Masker dat overeenstemt met het patroon in vergelijking (1)

Met een dergelijk masker kan een toestands-observatie tabel worden aangemaakt. Men kan dan aan elk masker een bepaalde kwaliteit toewijzen, die een afweging vormt van de graad van determinisme (via Shannon Entropie) en de graad van complexiteit die bij het masker horen. Verschillende definities voor de complexiteit van een masker zijn terug te vinden in hoofdstuk 2.

Het masker van Tabel 1 is maar één uit vele. De bedoeling van de systeemidentificatie is een masker te vinden dat een optimale kwaliteit heeft (hierbij kan de toestand-observatie tabel ook een rol spelen). Het zoeken naar dit optimaal masker gebeurt op een uitputtende (exhaustive) manier. Vertrekkende van ondiepe maskers worden alle mogelijke maskers gegenereerd en geëvalueerd (waarbij de cardinaliteit telkens wordt opgetrokken als alle maskers van een bepaalde cardinaliteit geëvalueerd zijn). Het masker dat de hoogste kwaliteit vertoont, wordt beschouwd als het ‘beste’ of optimale masker. Elk masker dat wordt gegenereerd in dit proces is een submasker van een ‘kandidaat’ of primair masker. Dit zoekproces is eindig omdat een primair (kandidaat) masker de zoekruimte beperkt. Dikwijls kan er eerder gestopt worden doordat de complexiteit van het voortgebrachte masker te hoog wordt, en de graad van determinisme nauwelijks of niet meer stijgt. Met het aldus gevonden optimale masker kan men nu op verschillende manieren voorspellingen doen. De eerste mogelijkheid steunt op het toestand-observatie model, de tweede is een dichtste-buren benadering (*k*-nearest neighbours)

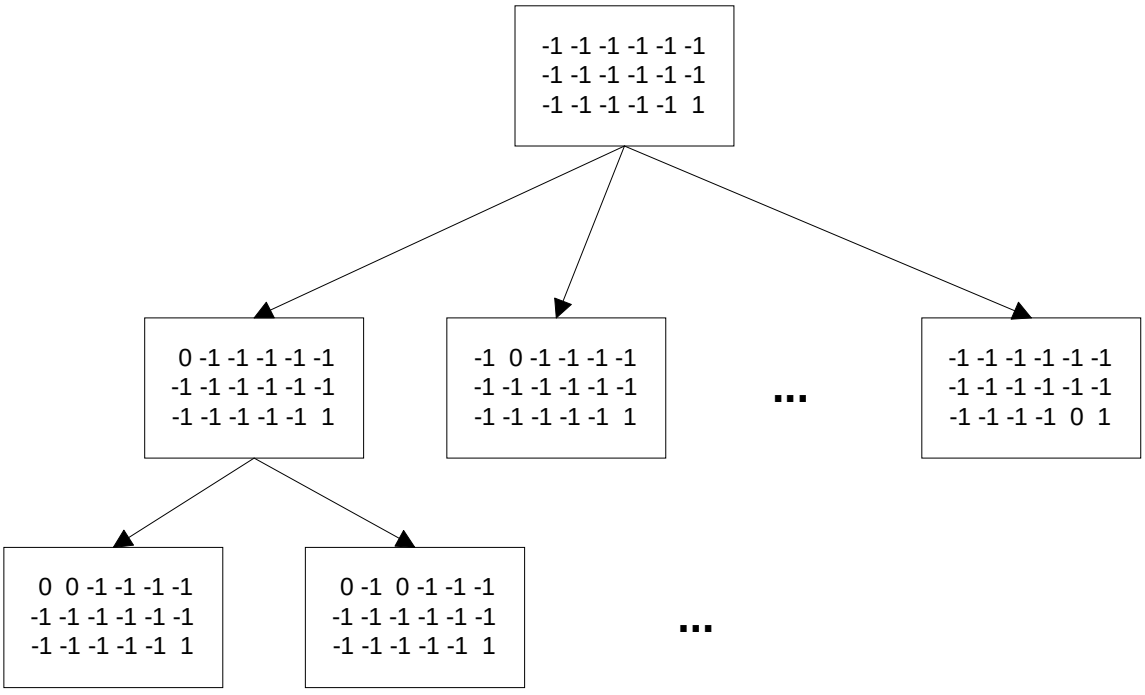
Hoofdstuk 3 introduceert toestandsmodellen. Aan de hand hiervan wordt het concept van een toestandstransitiematrix uitgelegd. Deze stochastische tijdsinvariante matrix is afgeleid van gediscretiseerde en gekwantiseerde observaties door hercodering. Dit suggereert dat een specifiek onderliggend stochastisch proces aan de oorsprong ligt. Dit proces, namelijk ‘verborgen Markov modellen’ (hidden Markov models), wordt hier uit de doeken gedaan. Doordat ‘*gecontroleerde* verborgen Markov modellen’ (nog) niet identificeerbaar zijn, werd een gegeneraliseerde toestand gedefinieerd waarbij men overgaat naar verborgen Markov modellen. Deze tonen duidelijk een formele overeenkomst met SAPS. Een nieuw probleemtype dat de opbouw van de belangrijke toestandsobservatie matrix in SAPS formaliseert, wordt in het kader van verborgen Markov modellen gedefinieerd en geanalyseerd. Dit probleemtype is, net als een ander verwant en bestaand type, analytisch oplosbaar. Het hoofdstuk eindigt met de conclusie dat een gelijkaardige problematiek de grondslag vormt voor de modelleerproblemen in verborgen Markov modellen en SAPS. De formalisatie toont aan dat correlatie tussen de argumenten in een patroon niet belangrijk is [Van Welden 20xx]. Bepaalde terminologie wordt duidelijk gedefinieerd.

Het eerste deel van hoofdstuk 4 betreft de introductie en rechtvaardiging van een, op heuristiek gebaseerde, suboptimale zoekstrategie [Van Welden and Vansteenkiste 1996]. De stelling dat het zoeken naar het *beste* masker op zich zinloos is, wordt opgeworpen en verdedigd. Eén van de argumenten hierbij is dat voor sommige (voldoende complexe) systemen er eenvoudigweg geen beste masker kan gevonden worden. Het bestaande algoritme vertoont namelijk een exponentiële complexiteit (2^n). Daardoor wordt het aantal mogelijke maskers dat moet onderzocht worden zo groot dat het rekenkundig niet meer haalbaar is om ze allemaal te evalueren. De bestaande uitputtende (exhaustive) zoekmethode volstaat dus geenszins om realistische complexe dynamische systemen te identificeren. Andere argumenten slaan op de subjectieve betekenis van 'beste' masker. Als gevolg hiervan wordt een nieuwe methode geïntroduceerd die niet meer het 'beste' masker zoekt (optimaal), maar een 'goed' masker (suboptimaal). Deze nieuwe zoekmethode is topdown en gebaseerd op een heuristiek (hill-climbing). Het corresponderende algoritme heeft een polynomiale complexiteit (n^2), wat uiteraard rekenkundig toelaat om een veel grotere maskerruimte te doorzoeken (zie ook Figuur 4). De bovengrens van de zoekruimte is gedefinieerd door een maximaal toegelaten masker (maximal allowable mask). Een voorbeeld van zo een masker vindt men in Tabel 2.

	invoeren					*uitvoer*
tijd	u_1	u_2	u_3	u_4	u_5	y
i-2	-1	-1	-1	-1	-1	-1
i-1	-1	-1	-1	-1	-1	-1
i	-1	-1	-1	-1	-1	1

Tabel 2 : Voorbeeld van een maximaal toegelaten masker

De boomstructuur gebruikt voor de hill-climbing methode is getekend in Figuur 4.



Figuur 4 : Gebruikte boom in de zoektocht voor het suboptimaal masker van Tabel 2

Het tweede deel van hoofdstuk 4 gaat dieper in op de implementatie-aspecten van de suboptimale masker zoekmethode. Een prototype is ontwikkeld in Smalltalk [4: ObjectShare 1999], vanwaar het programma zijn naam ontleent: SAPS-ST. Het prototype is functioneel verenigbaar met de bestaande SAPS-II versie[7] van Cellier, maar bevat uiteraard ook de nieuwe algoritmen om een suboptimaal masker te zoeken [Van Welden and Vansteenkiste 1994]. SAPS-ST is enorm gebruikersvriendelijk en hoeft geen programmering om gebruikt te worden. Vanwege zijn functie als prototype kunnen er ook andere hercoderingsalgoritmen eenvoudigweg ingevoegd worden en kan de maskerkwaliteitsfunctie eenvoudig gewijzigd worden via een 'drag and drop' mechanisme.

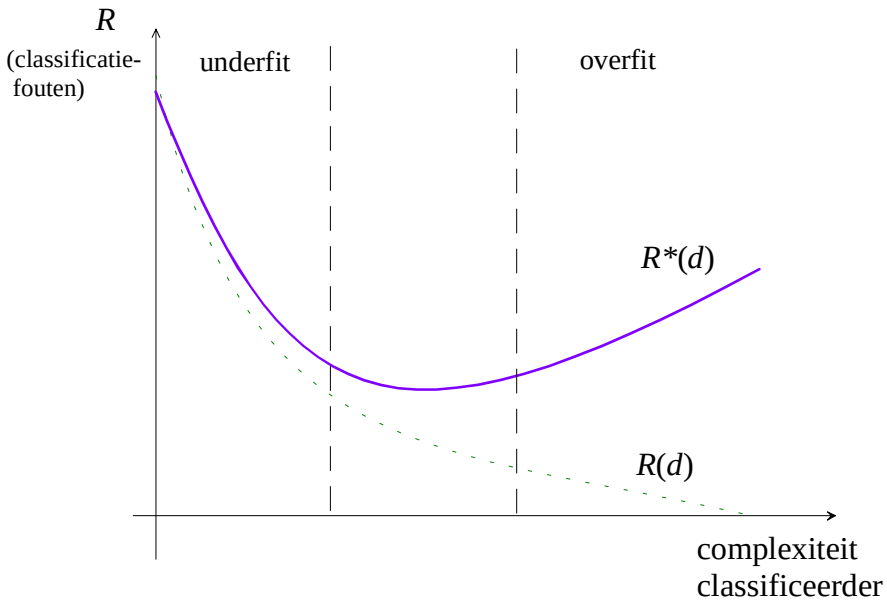
3.2 Deel II: Knowledge Discovery in Databases

De suboptimale zoekmethode, geïntroduceerd in deel I van de thesis, lijdt echter wel nog onder een aantal nadelen die ook van toepassing waren op de reeds bestaande uitputtende (exhaustive) methode. Hieronder bevinden zich: het efficiënt omgaan met ontbrekende waarden in de inputstroom van de observaties; het automatisch kwantiseren van de veranderlijken; het ongetekend (unbiased) valideren van het voorspellend vermogen van het model, enz. Een paradigma dat hier wel aan tegemoet komt, werd gevonden in het KDD domein door twee kernideeën, waarvan één reeds inherent in de oorspronkelijke GSPS methodologie aanwezig is, en de andere uit de suboptimale benadering over te nemen en in een nieuwe context te gieten. Het resultaat leent zich uitermate voor allerlei 'data-mining' methodes.

Hoofdstuk 5 geeft een kort overzicht van het KDD domein. Tevens wordt aangetoond dat data-mining een proces is in KDD dat in een grotere context dient geplaatst te worden. De knowledge-discovery (bottom-up) benadering wordt beschreven en gecontrasteerd met de hypothese benadering (topdown). Datawarehousing wordt besproken en ingekaderd in de levenscyclus van KDD, die stap voor stap bekeken wordt in hoofdstuk 5. Het machinaal leren (machine-learning) domein wordt kort toegelicht om er verhelderende concepten aan te ontlenen. Leren onder toezicht (supervised learning) zal blijken van toepassing te zijn op de systeemidentificatietaak als gesteld in het begin van dit doctoraat. Hoofdstuk 5 geeft verder een overzicht van de modellen die kunnen worden gebruikt voor deze vorm van leren, waarin classificeerders een belangrijke rol spelen. Zij leiden de boomclassificeerders in, die belangrijk zijn in de uiteindelijke toepassing van data-mining op SAPS-II. Dit past in het klassieke 'classificatie en voorspelling' probleemtype ('Classification and Prediction' problem type), waarin niet de snelheid van classificatie van belang is, maar wel de nauwkeurigheid. Nauwkeurigheid kan zowel op trainingsgegevens als op testgegevens gebaseerd zijn. De corresponderende classificatiefouten voor de trainingsobservaties ($R(d)$ in Figuur 5) zullen in aantal afnemen voor een meer complex model, maar het foutenaantal voor de testobservaties ($R^*(d)$ in Figuur 5) zal, na een initiële daling (underfit), terug een stijging (overfit) vertonen voor meer complexe modellen. Dit patroon is weergegeven in Figuur 5.

Wat betreft classificatiemodellen zal gestreefd worden naar begrijpbare (comprehensible) modellen. Zulke modellen vindt men in classificatie- en regressiebomen, die van pas komen in hoofdstuk 8 wanneer enkele voorbeelden de gemaakte sprong van GST naar KDD moeten verduidelijken.

[7] De SAPS versie van Cellier bevat wel nog toepassingen op het structureel niveau van GSPS



Figuur 5 : Resubstitutie versus werkelijke foutverhouding

Hoofdstuk 6 gaat dieper in op voornoemde classificatie- en regressiebomen. Dit hoofdstuk geeft de noodzakelijke achtergrondkennis om de toepassing van deze bomen in het kader van SAPS (hoofdstuk 8) te begrijpen. De onderliggende principes worden blootgelegd en verklaard. De verschillende constructiemethoden van classificatiebomen wordt behandeld. Er wordt verder uitgelegd hoe knopen in een boom zich splitsen, wanneer dit splitsen eindigt, en hoe een te ver gegroeide boom gesnoeid kan worden. De voordelen van dit snoeien worden toegelicht. Twee toonaangevende softwarepakketten worden beschreven. Elk pakket heeft zijn eigen implementatie van de basisprincipes die gelden voor classificatie- en regressiebomen. Het eerste softwarepakket, C4.5 [6: Quinlan 1993], heeft zijn wortels in het subdomein van machinaal leren. Omdat C4.5 niet verder wordt gebruikt in hoofdstuk 8, is de beschrijving ervan summier. Het andere softwarepakket, CART© [6: Salford 1999], afkomstig uit het subdomein van de statistiek, wordt wel verder in detail bekeken omdat het gebruikt wordt in hoofdstuk 8. Vooral de toepassing van surrogaatregels in de behandeling van 'missing values' en de mogelijkheid om regressiebomen te gebruiken gaven voor deze keuze de doorslag.

Hoofdstuk 7 behandelt de relatie tussen GST en KDD. Deze wordt beschreven op verschillende abstractieniveaus. Het eerste deel van hoofdstuk 7 vergelijkt de filosofische aspecten van GST en KDD. Hun respectievelijke levenscycli worden samengevoegd. Klemtoonverschillen tussen beide domeinen worden verder uitgelegd en aangetoond wordt hoe een wetenschappelijke kruisbestuiving kan plaatsvinden. Het tweede deel van dit hoofdstuk vergelijkt GSPS en KDD in meer detail waarbij het 'leren onder toezicht' paradigma beschouwd wordt. Een methodologische toepassing van data-mining technieken naar SAPS wordt summier besproken. Een meer gedetailleerde beschrijving met classificatie- en regressiebomen wordt uitgesteld tot het navolgende hoofdstuk 8.

Hoofdstuk 8 toont de toepasbaarheid van data-mining met behulp van classificatie- en regressiebomen voor SAPS. De sterke functionele gelijkenis tussen SAPS-ST en het relatief eenvoudig boomclassificatie-algoritme ID3 [6: Quinlan 1986] wordt bondig uitgelegd. Verder toont hoofdstuk 8 aan dat (manuele) hercodering met de nieuwe benadering niet meer nodig is. Classificatie- en regressiebomen doen dynamische discretisering. Deze maakt optimaal

gebruik van de aanwezige informatie in de veranderlijken. Regressiebomen, die kunnen opgebouwd worden met CART©, hebben het bijkomend voordeel dat ook de uitvoer niet gehercodeerd hoeft te worden. Het gebruik van regressiebomen laat toe om zeer complexe patronen te evalueren en in een voorspellingsmodel te gieten. De toepassingsvoorbeelden in de appendices A,B,C en D illustreren dit. CART© geeft tevens een rangschikking van veranderlijken. Deze rankschikking maakt een goede terugkoppeling naar SAPS, in de vorm van een primair masker, mogelijk. Het voorbeeld uit appendix C illustreert dit. Het gebruik van surrogaatsplitsingen laat een goede aanpak toe van observaties waarin waarden ontbreken. Dit wordt met een voorbeeld uit appendix C geïllustreerd. Tenslotte wordt ook het dichtste-buren (nearest neighbour) algoritme, aanwezig in SAPS-II, vergeleken met een nieuwe, krachtige en eenvoudige versie die geen hercodering behoeft van de testobservaties. De voorbeelden uit appendices A, B en C tonen aan dat de nieuwe methode effectief beter en sneller werkt. Tenslotte wordt ook geïllustreerd met een voorbeeld uit appendix B dat regels eenvoudig gegenereerd kunnen worden.

4 Besluit

De toegepaste wetenschappelijke bijdrage van dit doctoraatswerk is drievoudig.

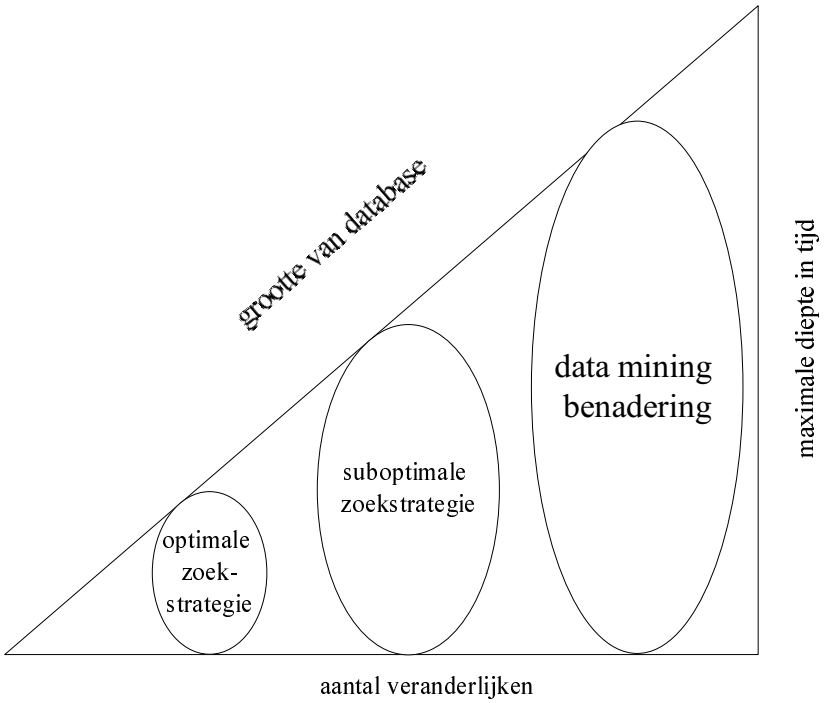
De eerste bijdrage situeert zich in het domein van GST. Het betreft de introductie van een suboptimale zoekstrategie, die het mogelijk maakt om naar meer complexe patronen te zoeken dan dat met SAPS-II mogelijk is. Men kan bijvoorbeeld afgeleiden in een masker bijvoegen en grotere tijdsconstanten bekijken. Niet alleen is een rechtvaardiging voor de nieuwe benadering gegeven, maar ze is ook geïmplementeerd in een software prototype. Dit prototype laat toe om verschillende zoekstrategieën uit te proberen en om de invloed van verschillende kwaliteitsfuncties te onderzoeken.

De bijdrage in het domein van KDD overstijgt de grenzen van het GSPS raamwerk. Een brug is gelegd van GST naar KDD. Een simpele (bestaande) transformatie, gecombineerd met een maximaal toelaatbaar masker, ligt aan de grondslag hiervan. Een reeks nieuwe methoden, die reeds aanwezig waren in KDD, komen nu aan bod als mogelijke kandidaatmethoden. Het is uiteraard onmogelijk om al deze kandidaatmethoden te onderzoeken op hun geschiktheid m.b.t. de probleemstelling beschreven in sectie 1. Daarom werden boomclassificeerders gebruikt omdat ze het meest aanleunen bij de suboptimale benadering in deel I van de thesis. De voorbeelden in de appendices werden allemaal met regressiebomen gemodelleerd. Uit de experimenten blijkt dat ze goede voorspellende eigenschappen bezitten. De hogere voorspellingsnauwkeurigheid t.o.v. de toestands-observatie voorspellingsmethode is overduidelijk. Zelfs tegenover de dichtste-nabuurmethoden houden ze goed stand wat betreft hun voorspellingsnauwkeurigheid. Dit is goed nieuws, want regressiebomen geven een compact globaal model dat snel kan voorspellen (eager evaluation). De dichtste-buurmethoden maken gebruik van lokale interpolatie. Ze hebben typisch de gehele gegevensverzameling nodig bij elke voorspelling (lazy evaluation).

Een derde bijdrage betreft de verbetering van de dichtste-nabuurmethoden. De nieuwe geïntroduceerde methoden zijn theoretisch beter onderbouwd, eenvoudiger, en sneller. Zij kunnen nog steeds samen met een (sub)optimaal masker of met een boomclassificeerder gebruikt worden. Deze laatste spelen dan de rol van *feature selectors*.

Deze thesis levert, naast de toegepaste wetenschappelijke, ook een theoretische bijdrage. Het formele verband met hidden Markov Models geeft meer inzicht in de onderliggende problematiek die gepaard gaat met niet-parametrische systeemidentificatie. De methodologische koppeling van GST en KDD creëert ruimte voor een vruchtbare uitwisseling van ideeën.

Tenslotte kunnen er nog richtlijnen gegeven worden in verband met de toepasbaarheid van de drie onderzochte methoden (optimale zoekstrategie, suboptimale zoekstrategie, en data-mining methode). Dit is voorgesteld in Figuur 6 die een *voorkeur* aangeeft voor hun gebruik. Hierbij dient wel opgemerkt te worden dat, bijvoorbeeld, regressiebomen ook nog toepasbaar zijn voor minder complexe systemen. Vooral hun externe validatiemethoden zijn interessant vanwege hun minder getekend (biased) karakter.



Figuur 6 : Toepasbaarheid van de verschillende strategieën

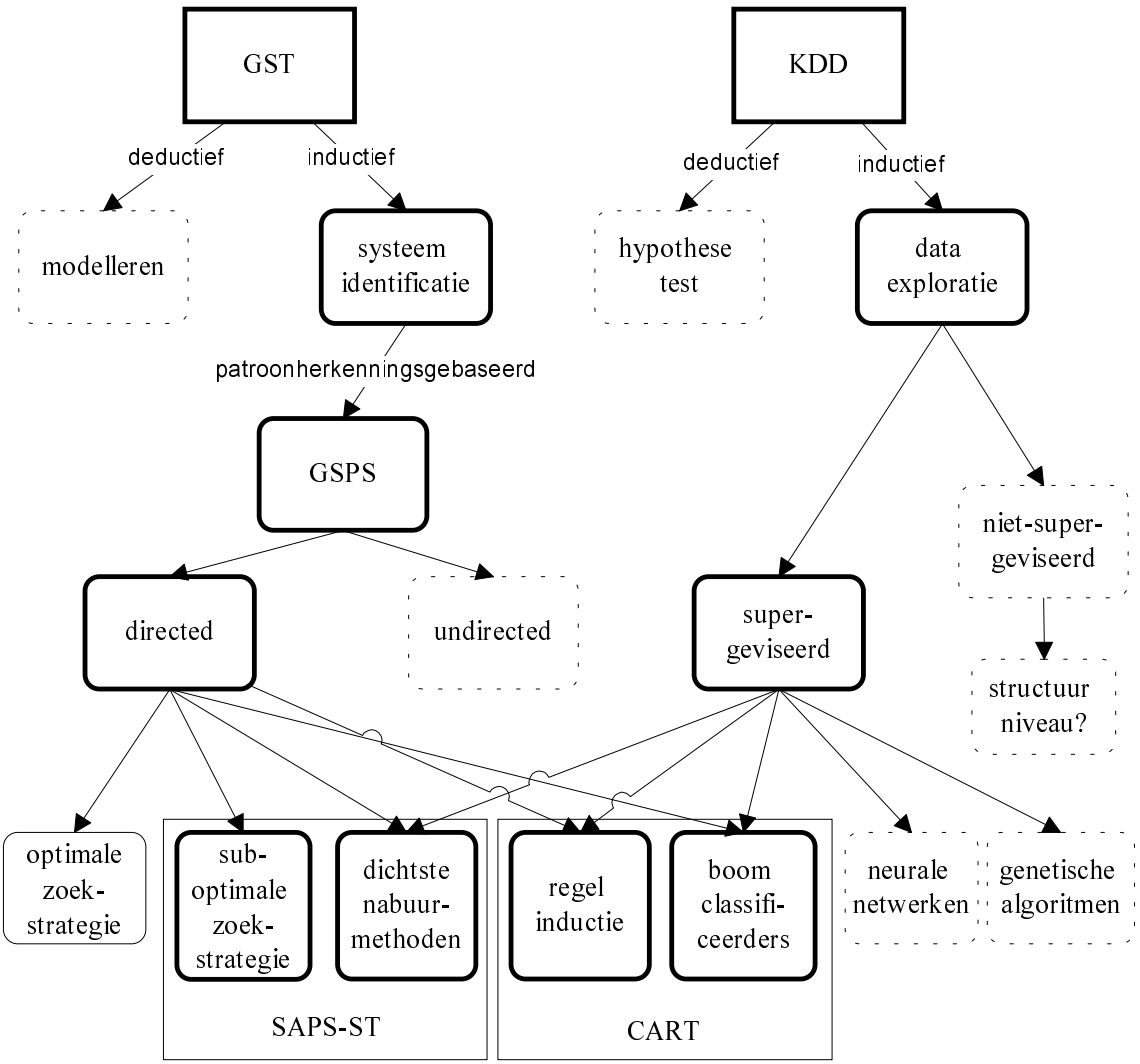
5 Richtlijnen voor verder onderzoek

Appendix A toonde aan dat het effect van de wijze van hercoding niet gemakkelijk in regels vast te leggen is. 'Fixed-recoding' gaf in het voorbeeld van appendix A een nauwkeuriger voorspelling dan 'Uniform-recoding'. Dit was niet te verwachten op theoretische gronden (hoofdstuk 2). Regressiebomen voldoen in ieder geval beter, maar daar wordt de validatie van het model gedaan via cross-validatie. Het is dus moeilijk om de vergelijking van 'fixed' en 'uniform' recoding door te trekken naar regressiebomen.

Dankzij de link met KDD kunnen nu nog andere data-mining methoden onderzocht worden op hun performantie voor SAPS. Dit wordt voorgesteld in Figuur 7. Neurale netwerken, genetische algoritmen, en niet-lineaire regressie behoren tot de mogelijkheden. Clustering technieken kunnen mogelijk worden toegepast om subsystemen te identificeren (structuurniveau in GSPS).

Zelfs als men bij boomclassificeerders blijft, dan nog zijn de mogelijkheden tot verder onderzoek legio. Het hercoderen van een uitvoer laat toe om bepaalde waarden als 'kritisch' te beschouwen (bijvoorbeeld: te hoge straling, te hoge of te lage temperatuur, enz.). Een kostgebaseerde classificeerder zou dan met een hogere nauwkeurigheid de ongewenste waardenzones kunnen voorspellen (of identificeren). Dit is maar één van de vele mogelijkheden tot verder onderzoek. Andere omvatten foutdetectie, betere discretisatie van veranderlijken, enz.

Men mag dus gerust stellen dat dit doctoraatswerk niet het einde, maar het begin is voor verder onderzoek.



Figuur 7 : Inductiemethoden in hun context